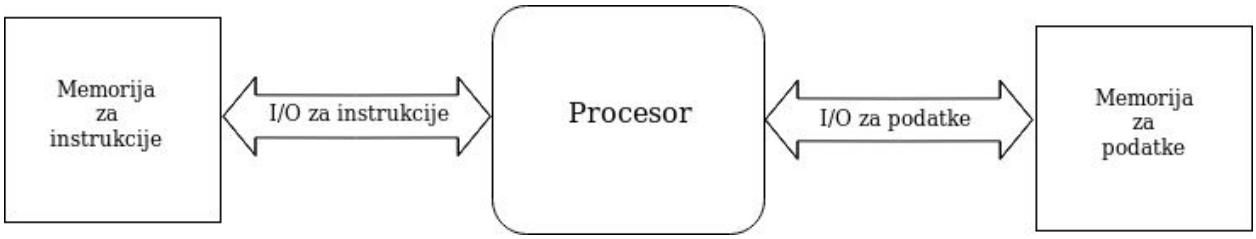
Vežba 2
Hardverska implementacija Single cycle RISC-V arhitekture

1. Uvod

Na prethodnim vežbama objašnjen je RISC-V set instrukcija i kako se na osnovu njega može napisati jednostavan program, dok će u nastavku biti objašnjeni osnovni principi i tehnike implementacije procesora koji podržava taj set instrukcija. No, da bi se oni razumeli neophodno je prvo razumeti funkcionalnost procesora počevši od njegove pojednostavljene predstave na visokom nivou abstrakcije.

Kao što je već pomenuto, procesor izvršava program predstavljen pomoću niza instrukcija, pri čemu se prilikom izvršavanja generišu određeni rezultati ili međurezultati. Da bi procesor mogao da izvršava program on mora da ima pristup memorijskim lokacijama na kojima je program smešten i mora da ima pristup memorijskim lokacijama u koje može da smešta, odnosno iz kojih može da preuzima određene podatke. Na slici 1 je prikazana konfiguracija koja ilustruje prethodno opisano:



Slika 1. Pojednostavljena predstava procesora na visokom nivou apstrakcije

Konfiguracija sa slike 1 ilustruje da procesor komunicira sa dve memorije. U prvu se smeštaju instrukcije, a u drugu podaci, pri čemu se komunikacija obavlja preko dva interfejsa:

- Interfejsa za komunikaciju sa memorijom za instrukcije, koji mora da obezbedi samo mogućnost čitanja
- Interfejsa za komunikaciju sa memorijom za podatke, koji mora da obezbedi i čitanje i pisanje.

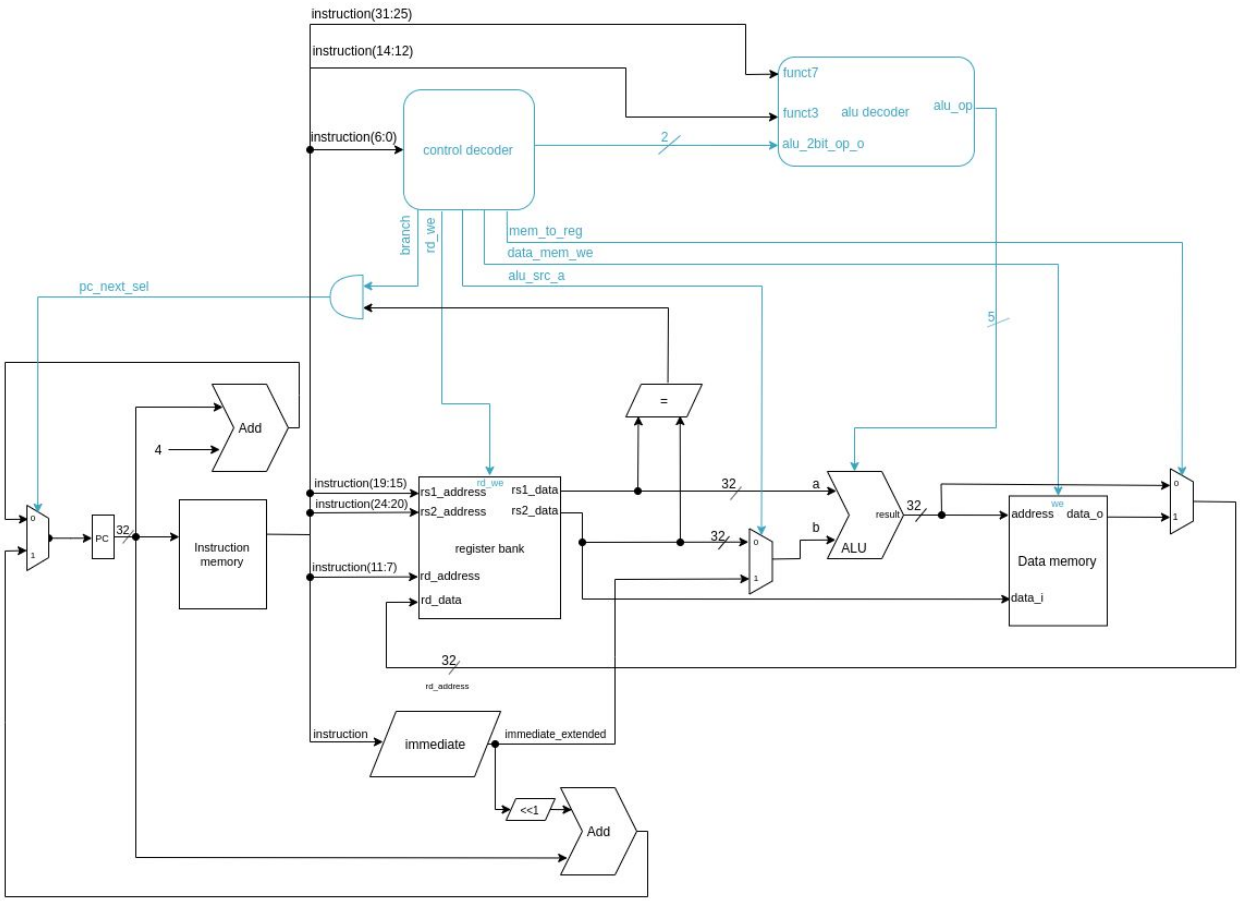
2. Single-cycle procesor

Prva i najjednostavnija implementacija procesora koja će biti objašnjena jeste Single-Cycle implementacija, specifična po tome da svaku instrukciju izvršava tačno jedan takt. Njena jednostavnost ima cenu, a to su lošije performanse u odnosu na druge implementacije (npr. Procesor sa protočnom obradom) i retko se koristi u savremenim procesorima, ali je korisna za edukativne svrhe jer predstavlja odličnu uvertiru u svet procesora.

Kako bi procesor izvršio određeni program, on mora da bude u stanju da prepozna svaku od instrukcija koje čine program i da izvrši određenu operaciju u zavisnosti od primljene instrukcije. No, kako je RISC-V set instrukcija veliki, na ovim i narednim vežbama biće implementiran samo osnovni podskup instrukcija, dok se čitaocu ostavlja da taj podskup instrukcija samostalno proširi. Instrukcije koje će biti implementirane su sledeće:

- Instrukcije koje omogućavaju pristup memoriji sa podacima: učitaj reč (***eng. Load word***) i skladišti reč (***eng. Store word***), sa oznakama ***lw i sw*** respektivno.
- Instrukcije koje izvršavaju aritmetičko logičke operacije: sabiranje, oduzimanje, logičko i i logičko ili, sa oznakama ***add, sub, and, or*** respektivno.
- Instrukcija uslovnog skoka ***(eng. Branch if equal)*** koja omogućava skok ukoliko je ispunjen uslov jednakosti, sa oznakom ***BEQ***.
- Instrukcija sabiranja sa konstantom ***(eng. add immediate),*** sa oznakom ***addi.***

Iako mali, ovaj podskup instrukcija će ilustrovati najvažnije principe korišćene u projektovanju procesora, dok će implementacija ostalih instrukcija zahtevati minimalne modifikacije. Na sledećoj slici je prikazana arhitekura procesora koja implementira prethodni set instrukcija:



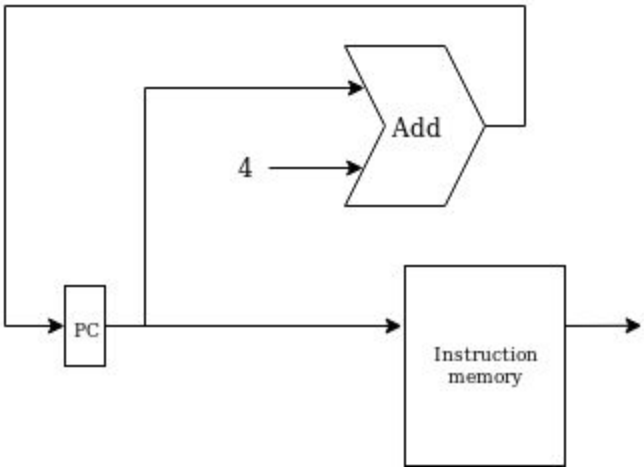
Slika 2. RISC-V procesor

Hardverska implementacija sa slike 2 je podeljena na dve velike celine: ***Controlpath***(obojeno plavo na slici) i ***Datapath***. ***Datapath*** se sastoji od osnovnih komponenti neophodnih da bi se izvršile prethodno navedene instrukcije i razlog korišćenja svake od njih biće objašnjen u nastavku. Controlpath predstavlja logiku za kontrolu komponenti u ***Datapath-u.*** U zavisnosti od instrukcije koju procesor trenutno izvršava, ***ControlPath*** generiše signale koji upravljaju komponentama u DataPath-u na takav način da se izvrši adekvatna operacija (npr. ALU će izvršiti sabiranje ukoliko je procesor prihvatio ADD instrukcija). ***Controlpath*** će detaljno biti objašnjen u nastavku. Na prethodnoj slici se može dobiti dojam da se memorija za instrukcije i memorija za podatke nalaze u datapath-u, odnosno da se nalaze unutar procesora, no, one su tu samo radi lakšeg vizuelnog prikaza i **ne pripadaju** arhitekturi procesora.

Bitno je naglastiti da će arhitektura procesora biti 32-bitna, odnosno sve instrukcije i podaci će biti predstavljeni sa 32 bita.

2.1 Datapath

Razuman početak implementacije datapath celine jeste da se razjasne glavne komponente neophodne da se izvrši svaka klasa RISC-V instrukcija. Te komponente su zadužene za prihvat instrukcije iz memorije i prikazane su na sledećoj slici:

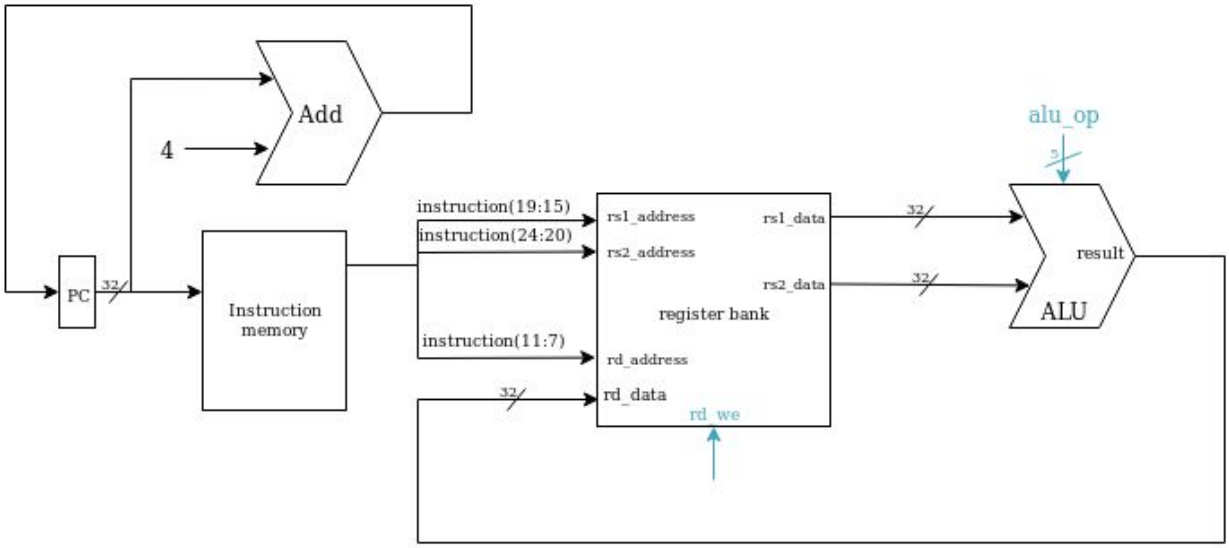


Slika 3. Inkrement programskog brojača i ekstrakcija instrukcije

Memorija za instrukcije je memorija sa asinhronim čitanjem i u njoj je smešten program koji treba da se izvrši. Koja instrukcije će biti ekstrahovana iz memorije zavisi od programskog brojača (PC na slici 3) koji generiše adresu na kojoj se nalazi naredna instrukcija. Sa slike 3 se može videti da se programski brojač uvek uvećava za 4 pomoću ***Add*** komponente koja predstavlja sabirač. Razlog za to je memorija za instrukcije koje je bajt adresibilna, odnosno četiri lokacije u memoriji predstavljaju jednu instrukciju i zato je naredna instrukcija pomerena u odnosu na prethodnu za 4 (instrukcija je 32-bitna, a to je 4 bajta). Takođe bitno je naglasiti da je programski brojač registar, što znači da će se naredna vrednost na njegovom izlazu pojaviti tek pri pojavi rastuće ivice takta, a kako je memorija za instrukcije memorija sa asinhronim čitanjem, pri svakoj promeni stanja programskog brojača promeniće se i izlaz memorije.

2.1.1 Implementacija ADD, SUB, OR, AND instrukcija

Nakon što je implementirana logika za prihvat naredne instrukcije počeće se sa proširivanjem arhitekture sa slike 3 dodatnim komponentama kako bi se podržale ***add, sub, or i and*** instrukcije.



Slika 4. Proširenje procesora kako bi podržao add, sub, or i and instrukcije

Slika 4 ilustruje da je neophodno dodati dve nove komponente: Registarsku banku i aritmetičko logičku jedinicu.

Registarska banka sadrži 32 registra širine 32 bita (arhitektura je 32-bitna) i uloga svakog od registara objašjena je na prethodnim vežbama. Komunikacija sa registarskom bankom odvija se preko dva interfejsa za čitanje i jednog interfejsa za upis podataka i razlog za to je format samih instrukcija. Primera radi, instrukcije koje pripadaju R tipu instrukcija moraju da pristupe na 3 lokacije istovremeno u registarskoj banci, sa dve treba da pročitaju i u treću treba da upišu rezultat operacije. Opis interfejsa je sledeći:

- Interfejsi za **čitanje** podaka sastoji se iz dva pristupa: rsN_address i rsN_data (slovo N predstavlja broj 1 ili 2). rsN_address je ulazni port koji predstavlja adresu sa koje treba pročitati podatak, dok je rsN_data izlazni port na kome će se pojaviti vrednost registra sa željene adrese. **Čitanje podataka iz memorije je asinhrono**.
- Interfejs za **upis** podataka se sastoji iz rd_address, rd_data i rd_we pristupa. Rd_address ulaz predstavlja adresu registra u koji se želi upisati podatak, rd_data predstavlja port preko koga se željena vrednost upisuje u registar i rd_we port predstavlja signal dozvole upisa u registarsku banku (Kontrolni signali

iz ***ControlPath*** celine kontrolišu njegovu vrednost). **Upis podataka u memoriju je sinhron.**

Sa slike 4 se može videti da je na svaki adresni ulaz registarske banke dovedeno 5 bita instrukcije, odnosno bitovi 19 do 15 su dovedeni na rs1_address port, bitovi 24 do 20 su dovedeni na rs2_address port i bitovi 11 do 7 su dovedeni na rd_address port. Da bi se razumelo zašto se baš ovi bitovi dovode na pomenute adresne ulaze, napraviće se osvrt na format R tipa instrukcije. On je prikazan na sledećoj slici:

funct7	rs2	rs1	funct3	rd	opcode
7 bita(31:25)	5 bita(24:20)	5 bita (19:15)	3 bita (14:12)	5 bita(11:7)	7 bita(6:0)

Slika 5. Format R tipa instrukcije

U ovom trenutku, pri kreiranju ***datapath*** celine, bitna su nam 3 polja prikazana na slici 5 i ta polja su rs2, rs1 i rd. Unutar njih je sadržana informacija o tome iz kojih registara registarske banke se čita (rs1 i rs2) i gde se smešta rezultat izvršene operacije (rd). Odnosno ta polja predstavljaju adrese registara u registarskoj banci. Na osnovu ove informacije jasno je zašto se prethodno pomenuti bitovi instrukcije dovode na tačno određene adresne ulaze registarske banke.

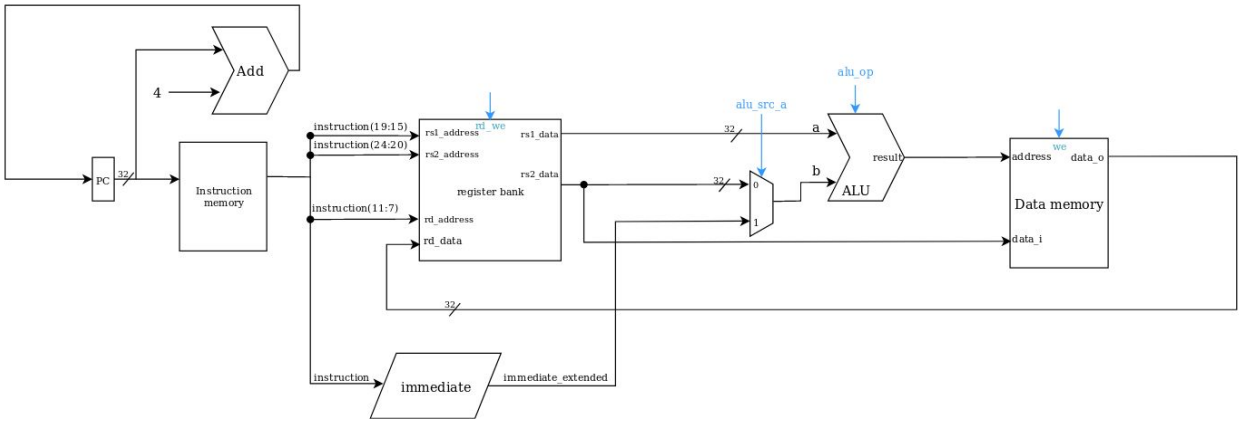
Druga komponenta koja je dodata jeste aritmetičko logička jedinica (na slici 4 označena sa ALU). Njena uloga može da se protumači iz imena, a to je obavljanje aritmetičko logičih operacija (sabiranje, oduzimanje, logičko ili, logičko i). Interfejs joj se sastoji iz sledećih linija:

- Dva ulazna porta za operande nad kojima treba da se izvrši aritmetičko logička operacija
- Ulaznog porta (***alu_op*** na slici 4), za izbor operacije koju treba izvršiti.
- Izlaznog porta (***result*** na slici 4) na kome se pojavljuje rezultat operacije.

Na slici 4 je prikazano da su na ulaze ALU jedinice dovedeni izlazi registarske banke (rs1_data i rs2_data), vrednost na ***alu_op*** ulazu kontroliše ***ControlPath*** (on određuje tip operacije koji treba se izvrši) i izlaz (result) je doveden rd_data ulaz registarske banke.

2.1.2 Implementacija instrukcije za upis u memoriju za podatke (sw)

Na slici 6 se može primetiti memorija za podatke koja prilikom implementiranja dosadašnjih instrukcija nije bila pominjana. Razlog je što instrukcije koje su do sada implementirane ne koriste memoriju za podatke, no kako je pristup ovoj memoriji neophodan, u nastavku će se implementirati instrukcije za čitanje i upis u nju. Prva koja će biti implementirana jeste instrukcija za upis (***sw***) u memoriju i na sledećoj slici biće prikazano proširenje slike 4 sa četiri dodatne komponene kako bi se pomenuta instrukcija podržala:



Slika 6. Proširenje procesora kako bi podržao ***SW*** instrukciju

Pre opisivanja pojedinačnih komponenti, napraviće se osvrt na S format instrukcija kome pripada ***sw:***

imm(11:5)	rs2	rs1	funct3	imm(4:0)	opcode
7 bita(31:25)	5 bita(24:20)	5 bita (19:15)	3 bita (14:12)	5 bita(11:7)	7 bita(6:0)

Slika 7. S format instrukcije

Polja koja su od interesa za datapath celinu su imm(11:5), rs2, rs1 i imm(4:0). Rs2 predstavlja adresu registra u registarskoj banci čiju vrednost treba upisati u memoriju za podatke. Imm(11:5) i imm(4:0) predstavljaju jednu celinu koja se zove immediate(konstanta), odnosno posmatraju se kao imm(11:0) i ta celina sabira se sa vrednošću na adresi rs1 u registarskoj banci kako bi se dobila adresa na koju je potrebno upisati podatak. Mašinski kod jedne takve instrukcije je sledeći:

Primer: sw a0, 4(a1).

Ovde a0 predstavlja registar rs2 čiju vrednost treba smestiti u memoriju, 4 predstavlja immediate polje ($000000000100_2 = 4_{10}$) i a1 predstavlja rs1. Odnosno

ukoliko je a0=5 i a1=4 to znači da će se na adresu 8 (a1 + immediate) memorije za podatke upisati vrednost registra na adresi 5 u registarskoj banci.

Sada se može preći na opis dodatnih komponenti na slici 6 neophodnih da bi se izvršila ***sw*** instrukcija. Memorija za podatke ima jako jednostavan interfejs koji se sastoji iz sledećih portova:

- ***Address*** port određuje sa koje adrese se čita, odnosno na koju adresu se piše.
- ***Data_i*** port predstavlja ulazni port za upis podataka. **Upis je sinhron**
- ***Data_o*** port predstavlja izlazni port čitanja podataka. **Čitanje je asinhrono**
- ***Data_we*** port predstavlja ulazni port dozvole upisa i kontrolisan je od strane signala koji potiče iz ***controlpath*** celine. Ukoliko je on na logičkoj jedinici pisanje je dozvoljeno, u suprotnom nije.

Sa slike 6 se može videti da je izlaz ALU jedinice doveden na adresni ulaz memorije sa podacima, razlog tome je sabiranje vrednosti registra na adresi rs1 i konstante (immediate) pri čemu je neophodno koristiti ALU jedinicu. Na ***data_i*** ulaz memorije sa podacima doveden je rs2_data izlaz registarske banke, jer je to informacija koju je potrebno upisati.

Sledeća komponenta je ***immediate***. Njena uloga je da proširi konstantu (immediate) sa 12 bita na 32, pri čemu se gornji biti proširuju nulama ili jedinicama u zavisnosti od najvišeg bita immediate polja kako bi se održao predznak konstante. Nezavisno od formata instrukcije, najviši bit immediate polja će se uvek poklapati sa najvišim bitom instrukcije (instruction(31)). Ukoliko je on logička jedinica immediate se do 32 bita proširuje sa logičkim jedinicama, u suprotnom se proširuje sa logičkim nulama.

11111111111111111111	***immediate(11:0)***
20 bita	12 bita

Slika 8. Proširivanje jedinicama ukoliko je instruction(31) = 1

00000000000000000000	***immediate(11:0)***
20 bita	12 bita

Slika 9. Proširivanje nulama ukoliko je instruction(31) = 0

Proširenje je neophodno uraditi kako bi ALU jedinica mogla da sabere immediate polje sa rs1_data izlazom registarske banke jer ona očekuje da oba ulazna operanda budu iste širine od 32 bita. Interfejs immediate komponente se sastoji od dva porta:

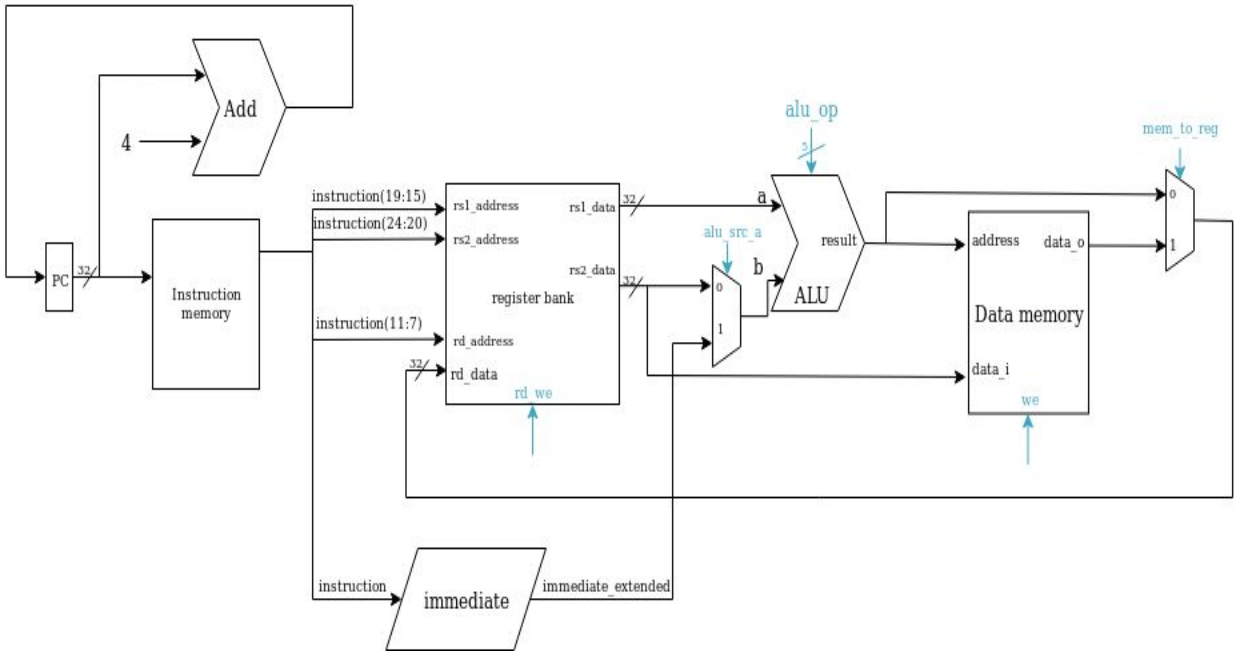
- ***Instruction*** predstavlja ulazni port preko koga se prima instrukcija.
- ***Immediate_extended*** izlazni port predstavlja prošireno immediate polje***.***

Immediate komponenta preko instruction porta dobija instrukciju iz koje izvlači immediate polje koje treba proširiti, ali takođe dobija i ***opcode*** polje. ***Opcode*** polje je neophodno jer ***immediate*** komponenta na osnovu njega zaključuje koji format instrukcije je u pitanju i shodno tome vrši proširivanje. Različiti formati zahtevaju različit način proširivanja. U slučaju sw instrukcije, koja ima S format, neophodno je izvršiti konkatanaciju polja imm(11:5) i polja imm (4:0), i onda ih proširiti do 32 bita.

Poslednja komponenta je multiplekser 2 na 1 i on je tu kako bi multipleksirao šta se dovodi na b ulaz ALU jedinice(***immediate_extend*** ili ***rs2_data***). Njega kontroliše ***controlpath*** preko ***alu_src_a*** ulaza.

2.1.3 Implementacija instrukcije za čitanje iz memoriju za podatke (lw)

Proširenje slike 6 za ***lw*** instrukciju je sledeće:



Slika 10. Proširenje procesora kako bi podržao ***LW*** instrukciju

Uloga ***lw*** instrukcije jeste da smesti podatak iz memorije za podatke u određeni registar u registarskoj banci i da bi se to izvršilo neophodno je ubaciti dodatne komponente. Da bi se razumela potreba za dodatnim multiplekserom napraviće se osvrt na I format instrukcija kome pripada lw:

imm(11:0)	rs1	funct3	rd	opcode
12 bita(31:20)	5 bita (19:15)	3 bita (14:12)	5 bita(11:7)	7 bita(6:0)

Slika 11. I format instrukcija

Polja koja su u ovom trenutku od interesa su imm(11:0), rs1 i rd. Rd predstavlja registar unutar registarske banke u koji će se smestiti podatak iz memorije, dok suma između rs1 i immediate polja predstavlja adresu u memoriji za podatke sa koje treba da se učita informacija.

U slučaju ove instrukcije immediate komponenta samo treba da proširi imm(11:0) polje instrukcije do 32 bita. Da li će se proširiti logičkim nulama ili jedinicama zavisi od imm(11) bita i to na isti način kao kod prethodne instrukcije.

Dodatna komponenta koja je neophodna jeste multiplekser 2 na 1 i on multipleksira šta će biti upisano u registarsku banku(rezultat alu jedinice ili podatak iz memorije). Njegov selekcioni ulaz je takođe kontrolisan od strane signala iz ***controlpath*** celine.

2.1.4 Implementacija ADDI instrukcije

Implementacija ADDI instrukcije ne zahteva dodatnu logiku u datapath celini. ADDI instrukcija pripada I formatu instrukcija (slika 11) zbog čega dodatne komponente nisu neophodne i sledeći primer će to ilustrovati:

Primer:

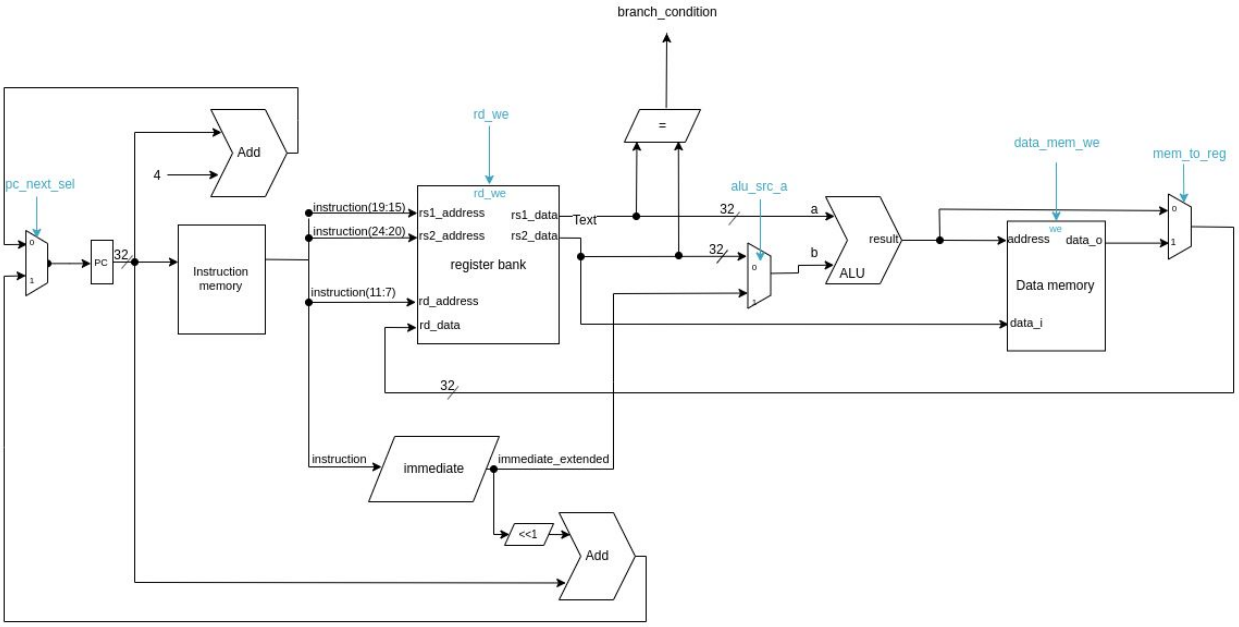
```
ADDI x10, x11, 10
```

Prethodna instrukcija sumira vrednost u registru X11 sa konstantom (immediate poljem) i rezultat smešta u registar x10. Na osnovu ovoga se zaključuje da je za sumiranje potrebna ALU jedinica, potrebno je multipleksirati “b” ulaz ALU jedinice[1] i potrebno je proslediti rezultat sumiranja u registarsku banku. Sve nabrojane komponente su već prisutne .

[1] Multipleksiranje je neophodno kako bi se na “b” ulaz ALU jedinice prosledila proširena vrednost immediate polja umesto rs2_data izlaza registarske banke.

2.1.5 Implementacija instrukcije uslovnog skoka (BEQ)

Slika 12 ilustruje proširenje slike 8 neophodnim komponentama kako bi procesor podržao instrukciju uslovnog skoka:



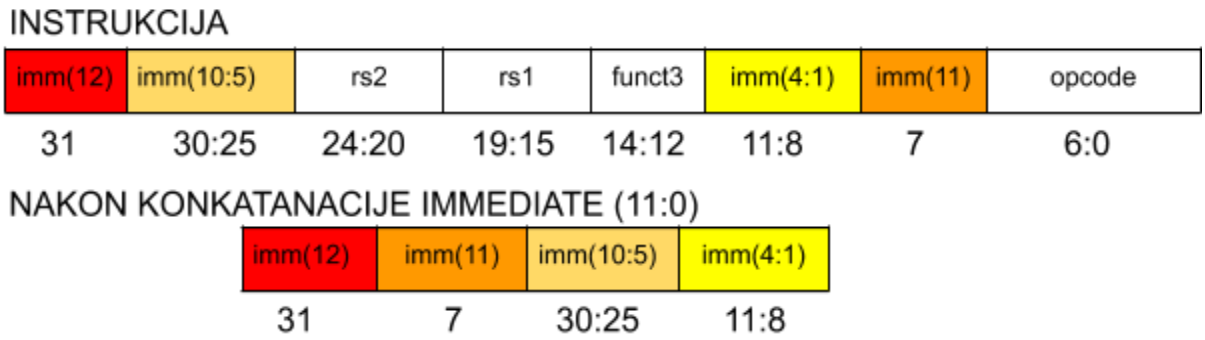
Slika 12. Proširenje procesora kako bi podržao ***BEQ*** instrukciju

Prilikom implementiranja svih instrukcija do sada, programski brojač (PC) se uvek uvećavao za konstantnu vrednost, odnosno instrukcije su uvek izvršavane sekvencijalno (jedna za drugom). Sada, dodavanjem instrukcijoe uslovnog skoka (BEQ), omogućava se skok na proizvoljnu instrukciju u memoriji ukoliko je uslov jednakosti ispunjen. Kako bi se razumele dodatne komponente napraviće se osvrt na SB format instrukcija kome BEQ pripada:

imm(12,10:5)	rs2	rs1	funct3	imm(4:1,11)	opcode
7 bita(31:25)	5 bita(24:20)	5 bita (19:15)	3 bita (14:12)	5 bita(11:7)	7 bita(6:0)

Slika 13. SB format instrukcije

Polja koja nisu od interesa u ovom trenutku su opcode i funct3. Rs1 i rs2 polja sadrže u sebi adrese registara koji se porede i od njihove jednakosti zavisi da li će se desiti skok ili ne. U ***immediate*** poljima (imm(12, 10:5) i imm(4:1, 11)) krije se informacija koliko bajtova treba skočiti do željene instrukcije. Da bi se ta informacija dobila ***immediate*** komponenta mora da izvrši konkatanaciju ***immediate*** polja instrukcije i nakon toga proširivanje do 32 bita. Konkatanacija se vrši spajanjem 31:25 i 11:7 bitova instrukcije, vodeći računa o poziciji bitova. Odnosno treba uraditi sledeće:



Slika 14. Ilustracija konkatanacije immediate bita

Proširivanje do 32 bita se radi kao i u prethodnim slučajevima, umeću se sa leve strane sve jedinice ili nule u zavisnosti od najvišeg bita instrukcije (instruction(31)):

11111111111111111111	***immediate(11:0)***
20 bita	12 bita

Slika 15. Proširivanje jedinicama ukoliko je instruction(31) = 1

00000000000000000000	***immediate(11:0)***
20 bita	12 bita

Slika 16. Proširivanje nulama ukoliko je instruction(31) = 0

Još jedna operacija koju je potrebno dodati jeste pomeranje proširene vrednost za jednu poziciju u levo i ako se pogleda slika 12, može se videti da je nakon ***immediate*** komponente dodat pomerač u levo[2]. Rezultat nakon pomeranja predstavljaće broj bajtova od trenutne vrednosti programskog brojača do naredne instrukcije koju treba izvršiti. Smisao pomeranja u levo za jednu poziciju jeste proširenje za duplo opsega instrukcija do kojih može da se skoči. Sledeći mašinski kod ilustruje zašto je pomeranje u levo neophodno:

[2] Na slici 12 nalazi se pomerač u levo, ali to je samo radi lakše ilustracije, u realnom slučaju on se nalazi unutar immediate komponente.

```
ADRESA      INSTRUKCIJA:
00:         addi a0 , a0, 10
04:         addi a1 , a1, 10
08:         beq a0, a1, L1
12:         add a2, a1, a1
16:         L1:sub a2, a1, a0
```

Slika 17. Primer mašinskog koda

U ovom slučaju BEQ instrukcija treba da skoči na 16 bajt, odnosno treba da preskoči jednu instrukciju. Kada se BEQ instrukcija pretvori u binarni kod dobija se sledeće :

0000000	01011	01010	000	01000	1100011
7 bita(31:25)	5 bita(24:20)	5 bita (19:15)	3 bita (14:12)	5 bita(11:7)	7 bita(6:0)

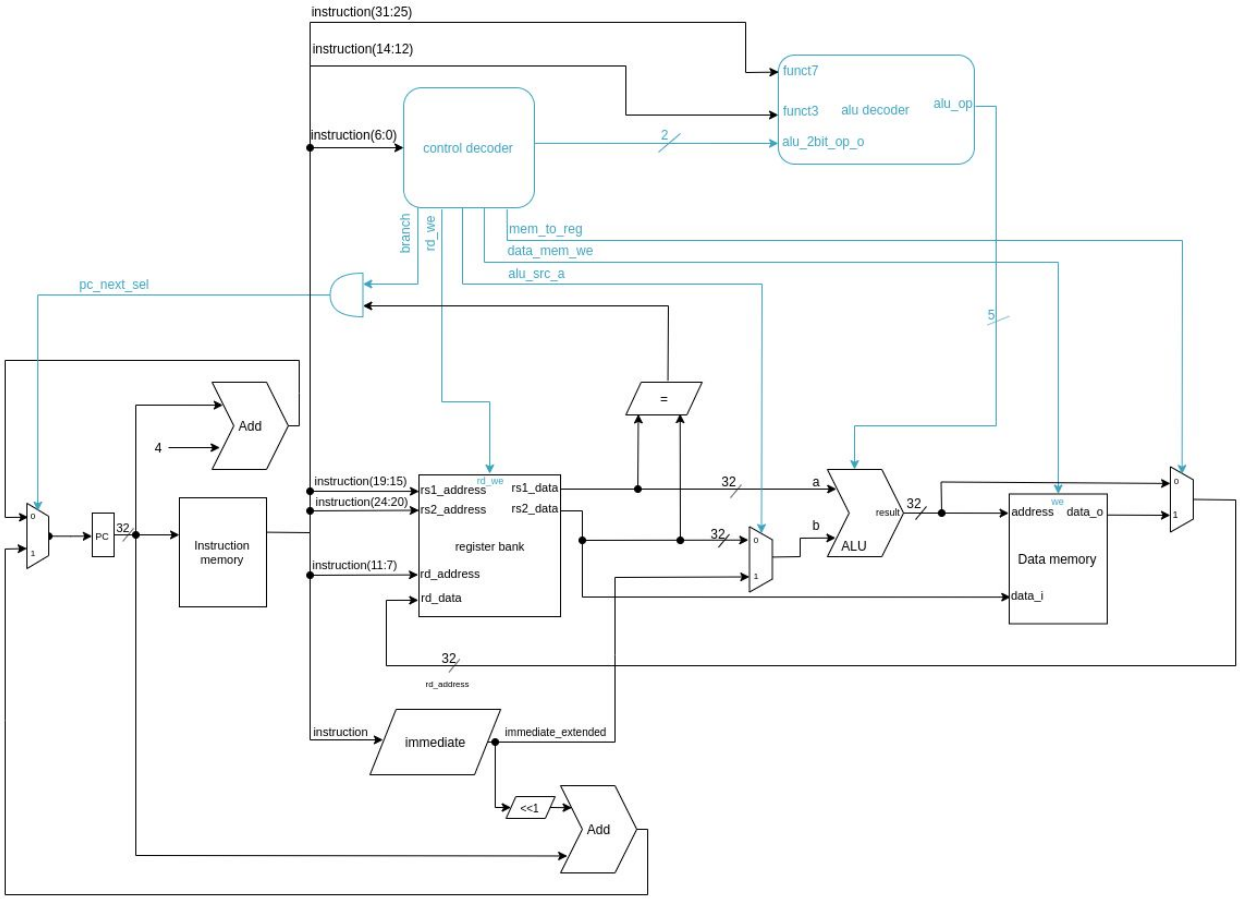
Slika 18. Primer BEQ instrukcije

Nakon što se izvrši ekstrahovanje ***immediate*** polja instrukcije i izvrši se prethodno opisani postupak konkatanacije i proširivanja, dobija se vrednost $00000004_{16} = 4_{10}$. Kako je potrebno preskočiti 8 bajtova a ne 4, neophodno je izvrišiti pomeranje u levo za jednu poziciju(ekvivalentno množenju sa 2).

Nakon pomeranja u levo za jednu poziciju, ta vrednost se sabira sa tretnutnom vrednošću programskog brojača i trebalo bi da se prosledi do ulaza PC registra. Da li će se proslediti ili ne zavisi od multipleksera koji se nalazi ispred PC registra čiji je selekcioni ulaz kontrolisan od strane ***pc_next_sel*** kontrolnog signala. Njega generiše ***controlpath*** i da bi to ispravno uradio neophodna mu je informacija o tome da li su operandi koje se porede (primer: a0 i a1 na slici 17) jednaki. Tu informaciju dobija u vidu statusnog signala ***branch_condition*** (slika 12) koji je izlaz iz komparatora koji poredi prethodno pomenute operande.

2.2 Controlpath

Nakon kreiranja ***datapath*** celine neophodno je kreirati ***controlpath*** koji će kontrolisati prethodno pomenute kontrolne signale. Ubacivanjem komponenti koje pripadaju controlpath celini dobija se sledeća slika:



Slika 18. Proširenje procesora sa ***controlpath*** komponentama

Sa slike 18 se može videti da ***controlpath*** čine dve komponente: ***Control decoder*** i ***alu decoder***[3].

Alu decoder komponenta je zadužena za kontrolisanje ALU jedinice, odnosno ona određuju koju instrukciju ALU treba da izvrši. Njen interfejs se sastoji iz sledećih portova:

- alu_2bit_op: 2-bitni ulazni port kontrolisan od strane ***control decoder*** komponente..

[3] Razdvajanje na dve komponente nije neophodno, odnosno ***control decoder*** i ***alu decoder*** su mogli biti jedna komponenta.

- funct3: 3-bitni ulazni port preko koga se primaju 3 bita instrukcije koji predstavljaju funct3 polje (instruction(14:12)).
- funct7: 7-bitni ulazni port preko koga se prima 7 bita instrukcije koji predstavljaju funct7 polje (instruction(31:25)).
- alu_op: 5-bitni izlazni port koji generiše kontrolne signale za upravljanje ALU jedinicom.

Sledeća slika ilustruje kako se u zavisnosti od ulaznih portova generiše ***alu_op*** kontrolni signal:

Alu_2bit_op ulaz		Funct7 ulaz							Funct3 ulaz			Alu_op izlaz
1	0	6	5	4	3	2	1	0	2	1	0	
0	0	X	X	X	X	X	X	X	X	X	X	00010
X	1	X	X	X	X	X	X	X	X	X	X	00110
1	X	0	0	0	0	0	0	0	0	0	0	00010
1	X	0	1	0	0	0	0	0	0	0	0	00110
1	X	0	0	0	0	0	0	0	1	1	1	00000
1	X	0	0	0	0	0	0	0	1	1	0	00001

Slika 19. Tablica istinosti za alu_op izlaz

Slika 20 daje informaciju koju operaciju ALU treba da izvrši u zavisnosti od kombinacije koju na svom izlazu generiše ***alu_op***.

Alu_op	operacija
00000	Logičko “I”
00001	Logičko “ILI”
00010	Sabiranje
00110	Oduzimanje

Slika 20. Veza ***alu_op*** kontrolnih signala i operacije koje ALU izvršava

Control_decoder komponenta je zadužena za generisanje kontrolnih signala kojima se kontrolišu sve ostale komponente u ***datapath*** celini i njen interfejs se sastoji iz sledećih portova:

- ***opcode***: ulazni port preko koga se prima donjih 7 bita instrukcije (***instruction(6:0)***).

- ***branch:*** Izlazni port koji se postavlja na visok logički nivo ukoliko je iz instrukcione memorije zahvaćena ***BEQ*** instrukcija.
- ***rd_we***: izlazni port koji se postavlja na visok logički nivo ukoliko instrukcija prihvaćena iz instrukcione memorije upisuje podatak u registarsku banku.
- ***alu_src_a:*** izlazni port koji kontroliše multiplekser ispred aritmetičko logičke jedinice i koji se postavlja na visok logički nivo ukoliko na “b” ulaz ALU jedinice treba da se prosledi ***immediate_extended*** signal.
- ***data_mem_we***: izlazni port koji se postavlja na visok logički nivo ukoliko instrukcija prihvaćena iz instrukcione memorije upisuje podatak u memoriju za podatke.
- mem_to_reg: izlazni port koji kontroliše multiplekser ispred memorije za podatke i koji se postavlja na visok logički nivo ukoliko je u registarsku banku potrebno upisati podatak iz memorije za podatke, a na nizak ukoliko je potrebno upisati izlaz ALU jedinice
- alu_2bit_op: 2-bitni izlazni port koji se prosleđuje ***alu_decoder*** komponenti, i na osnovu koga ona zaključuje koju operaciju ALU jedinica treba da izvrši.

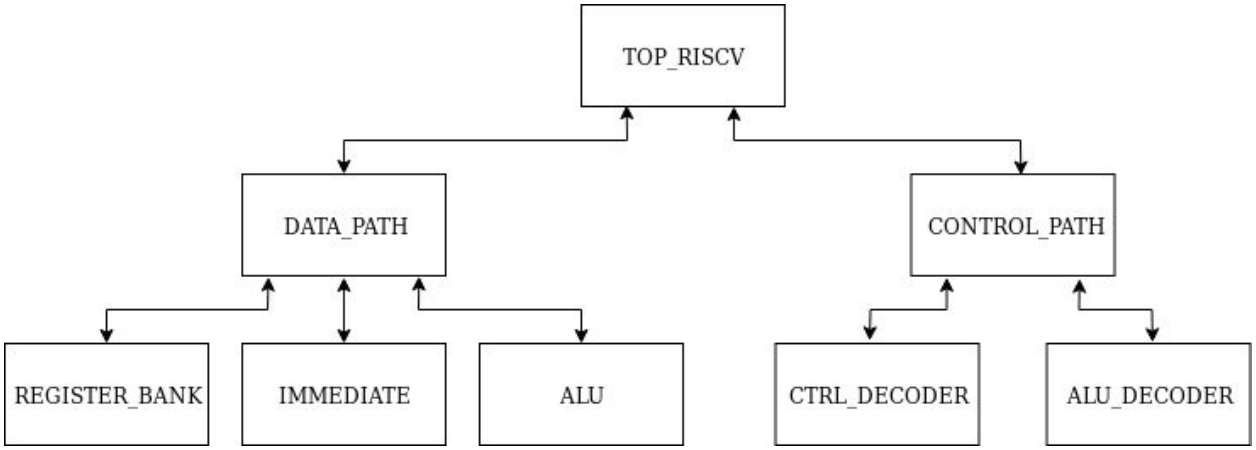
Sledeća slika ilustruje koji kontrolni signali se generišu u zavisnosti od tipa instrukcije (vrednosti ***opcode*** polja):

	Ime signala	R-format	lw	sw	beq	addi
Ulazi	opcode(6)	0	0	0	1	0
	opcode(5)	1	0	1	1	0
	opcode(4)	1	0	0	0	1
	opcode(3)	0	0	0	0	0
	opcode(2)	0	0	0	0	0
	opcode(1)	1	1	1	1	1
	opcode(0)	1	1	1	1	1
Izlazi	Alu_src	0	1	1	x	1
	Mem_to_reg	0	1	x	0	0
	rd_we	1	1	0	0	1
	data_mem_we	0	0	1	0	0
	branch	0	0	0	1	0
	Alu_2bit_op(1)	1	0	0	0	1
	Alu_2bit_op(0)	0	0	0	1	1

Slika 21. Veza Kontrolnih signala i opcode polja instrukcije

3 VHDL implementacija

Do sada je samo opisano i ilustrovano kako implementirati procesor sa određenim setom instrukcija, dok će se u nastavku pomoću vhdl jezika za opis hardvera izvršiti implementacija. Na sledećoj slici je prikazana hijerarhijska predstava procesora, odnosno kako su prethodno opisane celine međusobno povezane.



Slika 22. RISCV hijerarhija

3.1 REGISTER_BANK

Interfejs registarske banke je već opisan (pogledati 2.1.1) i sledeći kodni listing prikatuje njenu vhdl implementaciju. Ako se pogleda kod može se primetiti da se upis u registarsku banku vrši na opadajuću ivicu taktnog signala. Razlog za to biće objašnjen na sledećim laboratorijski vežbama prilikom implementacije procesora sa protočnom obradom.

```
daZjYjq a]]]Å
mk] a]]]Çkl\ëdg_a[ëúúÿýÇYddÅ
mk] a]]]Çfme]ja[ëkl\ÇYddÅ
ÈÈãããããããããããããã/0)3 -/$5,!ããããããããããããããããããã
ÈÈ2]_aklYjkcY ZYfcY kY \nY afl]j^]bkY rY [alYfb]
ÈÈhg\YlYcY a b]\fae afl]j^]bkge rY mhak hg\YlYcYÇ
ÈÈ"jgb j]_aklYjY m ZYf[a b] üûÇ
ÈÈ7)$4( b] hYjYe]lYj cgba g\j]\bmb] kajafm hg\YÈ
ÈÈ\YlY m j]_akljaeY
ÈÈãããããããããããããããããããããããããããããããããããããããããã
]flalq j]_akl]jëZYfc ak
   _]f]ja[ Ý7)$4( Æ hgkalan] Æđ üûÞÅ
```

```
   hgjl Ý[dc           Æ af  kl\ëdg_a[Å
        j]k]l          Æ af  kl\ëdg_a[Å
        ÈÈ )fl]j^]bk ú rY [alYfb] hg\YlYcY
        jkúëY\\j]kkëa Æ af  kl\ëdg_a[ën][lgjÝý \goflg ùÞÅ
        jkúë\YlYëg    Æ gml kl\ëdg_a[ën][lgjÝ7)$4( È ú \goflg ùÞÅ
        ÈÈ )fl]j^]bk û rY [alYfb] hg\YlYcY
        jkûëY\\j]kkëa Æ af  kl\ëdg_a[ën][lgjÝý \goflg ùÞÅ
        jkûë\YlYëg    Æ gml kl\ëdg_a[ën][lgjÝ7)$4( È ú \goflg ùÞÅ
        ÈÈ )fl]j^]bk rY mhak hg\YlYcY
        j\ëo]ëa       Æ af  kl\ëdg_a[Å  ÈÈ hgjl rY \grngdm mhakY
        j\ëY\\j]kkëa  Æ af  kl\ëdg_a[ën][lgjÝý \goflg ùÞÅ
        j\ë\YlYëa     Æ af  kl\ëdg_a[ën][lgjÝ7)$4( È ú \goflg ùÞÞÅ

]f\ ]flalqÅ

Yj[`al][lmj] "]`YnagjYd g^ j]_akl]jëZYfc ak
   lqh] j]_ëZYfc ak YjjYq Ýù lg üúÞ g^ kl\ëdg_a[ën][lgjÝüú \goflg ùÞÅ
   ka_fYd j]_ëZYfcëk Æ j]_ëZYfcÅ

Z]_af
   ÈÈ 3af`jgfa mhak hg\YlYcY m j]_aklYjlkcm ZYfcm
   j]_ëZYfcëojal] Æ hjg[]kk Ý[dcÞ ak
   Z]_af
      a^ Ý^Yddaf_ë]\_]Ý[dcÞÞl`]f
         a^ Ýj]k]l đ ìùìÞl`]f
            j]_ëZYfcëk Ēđ Ýgl`]jk đē Ýgl`]jk đē ìùìÞÞÅ
         ]dka^ Ýj\ëo]ëa đ ìúìÞ l`]f
            j]_ëZYfcëkÝlgëafl]_]jÝmfka_f]\Ýj\ëY\\j]kkëaÞÞÞ Ēđ j\ë\YlYëaÅ
         ]f\ a^Å
      ]f\ a^Å
   ]f\ hjg[]kkÅ
ÈÈ !kaf`jgfg [alYfb] hg\YlYcY ar j]_aklYjlkc] ZYfc]Ç 0g   ÈÈkh][a^acY[aba fmdla
j]_aklYj b] mn]c ùÇ
   j]_ëZYfcëj]Y\ Æ hjg[]kk ÝjkúëY\\j]kkëaÄ jkûëY\\j]kkëaÄ j]_ëZYfcëkÞ ak
   Z]_af

      a^Ýlgëafl]_]jÝmfka_f]\ÝjkúëY\\j]kkëaÞÞ đ ùÞ l`]f
         jkúë\YlYëg Ēđ kl\ëdg_a[ën][lgjÝlgëmfka_f]\ÝùÄ 7)$4(ÞÞÅ
      ]dk]
         jkúë\YlYëg Ēđ j]_ëZYfcëkÝlgëafl]_]jÝmfka_f]\ÝjkúëY\\j]kkëaÞÞÞÅ
      ]f\ a^Å

      a^Ýlgëafl]_]jÝmfka_f]\ÝjkûëY\\j]kkëaÞÞ đ ùÞ l`]f
         jkûë\YlYëg Ēđ kl\ëdg_a[ën][lgjÝlgëmfka_f]\ÝùÄ 7)$4(ÞÞÅ
      ]dk]
         jkûë\YlYëg Ēđ j]_ëZYfcëkÝlgëafl]_]jÝmfka_f]\ÝjkûëY\\j]kkëaÞÞÞÅ
      ]f\ a^Å
```

```
   ]f\ hjg[]kkÅ

]f\ Yj[`al][lmj]Å
```

3.2 IMMEDIATE

Kao i kod registarske banke, interfejs immediate komponente i njena funkcionalnost su već opisani (pogledati 2.1.2) i sledeći kodni listing je samo vhdl implementacija. Može se primetiti da je deo koda zakomentarisan. Ukoliko se čitalac odluči da proširi set instrukcija koji trenutno podržava ova implementacija RISC-V procesora te linije može otkomentarisati.

```
daZjYjq a]]]Å
mk] a]]]Çkl\ëdg_a[ëúúÿýÇYddÅ
mk] a]]]Çfme]ja[ëkl\ÇYddÅ


]flalq aee]\aYl] ak
   hgjl Ýafkljm[lagfëa        Æ af  kl\ëdg_a[ën][lgj Ýüú \goflg ùÞÅ
         aee]\aYl]ë]pl]f\]\ëg Æ gml kl\ëdg_a[ën][lgj Ýüú \goflg ùÞÞÅ
]f\ ]flalqÅ


Yj[`al][lmj] "]`YnagjYd g^ aee]\aYl] ak
   ka_fYd gh[g\]           Æ kl\ëdg_a[ën][lgjÝÿ \goflg ùÞÅ
   ka_fYd afkljm[lagfëlqh] Æ kl\ëdg_a[ën][lgjÝû \goflg ùÞÅ
   ÈÈka_fYd ^mf[lü           Æ kl\ëdg_a[ën][lgjÝû \goflg ùÞÅ
   ka_fYd ]pl]fkagf        Æ kl\ëdg_a[ën][lgjÝúĂ \goflg ùÞÅ

   [gfklYfl jëlqh]ëafkljm[lagf Æ kl\ëdg_a[ën][lgjÝû \goflg ùÞ Æđ íùùùíÅ
   [gfklYfl aëlqh]ëafkljm[lagf Æ kl\ëdg_a[ën][lgjÝû \goflg ùÞ Æđ íùùúíÅ
   [gfklYfl këlqh]ëafkljm[lagf Æ kl\ëdg_a[ën][lgjÝû \goflg ùÞ Æđ íùúùíÅ
   [gfklYfl Zëlqh]ëafkljm[lagf Æ kl\ëdg_a[ën][lgjÝû \goflg ùÞ Æđ íùúúíÅ
   ÈÈãã/lcge]flYjakYla mcgdacg b] lg hglj]Zfg hjadacge hjgkajanYfbY k]lY
afkljmc[abYãã

   ÈÈ[gfklYfl mëlqh]ëafkljm[lagf Æ kl\ëdg_a[ën][lgjÝû \goflg ùÞÆđ íúùùíÅ
   ÈÈ[gfklYfl bëlqh]ëafkljm[lagf Æ kl\ëdg_a[ën][lgjÝû \goflg ùÞÆđ íúùúíÅ
   ÈÈ[gfklYfl k`Yeläafkljm[lagf  Æ kl\ëdg_a[ën][lgjÝû \goflg ùÞÆđ íúúùíÅ
   ÈÈ[gfklYfl ^]f[]ë][Yddë]Zj]Yc Æ kl\ëdg_a[ën][lgjÝû \goflg ùÞÆđ íúúúíÅ

Z]_af
```

```
   gh[g\]    Ēđ afkljm[lagfëaÝÿ \goflg ùÞÅ
   ]pl]fkagf Ēđ Ýgl`]jk đē afkljm[lagfëaÝüúÞÞÅ
   ÈÈ^mf[lü    Ēđ afkljm[lagfëaÝúý \goflg úûÞÅ

   ÈÈ m g\fgkm fY gh[g\] hjgfY\ba afkljmc[abm
   hjg[]kk Ýgh[g\]Þ ak
   Z]_af
      [Yk] gh[g\]Ýÿ \goflg ûÞ ak
         o`]f íùúúùùí đē
            afkljm[lagfëlqh] Ēđ jëlqh]ëafkljm[lagfÅ
         o`]f íùùúùùí đē
            afkljm[lagfëlqh] Ēđ aëlqh]ëafkljm[lagfÅ
         o`]f íùúùùùí đē
            afkljm[lagfëlqh] Ēđ këlqh]ëafkljm[lagfÅ
         o`]f íúúùùùí đē
            afkljm[lagfëlqh] Ēđ Zëlqh]ëafkljm[lagfÅ
         o`]f gl`]jk đē
            afkljm[lagfëlqh] Ēđ jëlqh]ëafkljm[lagfÅ
      ]f\ [Yk]Å
   ]f\ hjg[]kkÅ

   ÈÈ fY gkfgnm afkljmc[ab] ar hj]l`g\fg_ hjg[]kYÄ ar\ngba a hjgkaja
ÈÈcgfklYflmÝaee]\aYl] hgdb]Þ fY üû ZalY
   hjg[]kk Ýafkljm[lagfëaÄ afkljm[lagfëlqh]Ä ]pl]fkagfÞ ak
   Z]_af
      [Yk] afkljm[lagfëlqh] ak
         o`]f aëlqh]ëafkljm[lagf đē
            aee]\aYl]ë]pl]f\]\ëg Ēđ ]pl]fkagf µ afkljm[lagfëaÝüú \goflg ûùÞÅ
         o`]f Zëlqh]ëafkljm[lagf đē
            aee]\aYl]ë]pl]f\]\ëg Ēđ ]pl]fkagfÝúā \goflg ùÞ µ afkljm[lagfëaÝüúÞ µ
afkljm[lagfëaÝĀÞ µ
                                    afkljm[lagfëaÝüù \goflg ûþÞ µ afkljm[lagfëaÝúú
\goflg āÞ µ ìùìÅ
         o`]f këlqh]ëafkljm[lagf đē
            aee]\aYl]ë]pl]f\]\ëg Ēđ ]pl]fkagfÝúĂ \goflg ùÞ µ afkljm[lagfëaÝüú
\goflg ûþÞ µ afkljm[lagfëaÝúú \goflg ĀÞÅ
         o`]f gl`]jk đē
            aee]\aYl]ë]pl]f\]\ëg Ēđ Ýgl`]jk đē ìùìÞÅ
      ]f\ [Yk]Å
   ]f\ hjg[]kkÅ
]f\ Yj[`al][lmj]Å
```

3.3 ALU

Sledeći kodni listing predstavlja vhdl implementaciju ALU jedinice. Operacije koje trenutno podržava ALU jedinica su samo aritmetičko logičke operacije koje do sada implementirane instrukcije zahtevaju. Ukoliko se čitalac odluči da proširi set instrukcija, moraće da modifikuje ALU jedinicu.

```
,)"2!29 a]]]Å
53% a]]]Çkl\ëdg_a[ëúúÿýÇ!,,Å
53% a]]]Çfme]ja[ëkl\Ç!,,Å
mk] a]]]ÇeYl`ëj]YdÇYddÅ
mk] ogjcÇYdmëghkëhc_ÇYddÅ


%.4)49 !,5 )3
   '%.%2)#Ý
      7)$4( Æ .!452!, Æđ üûÞÅ
   0/24Ý
      Yëa    Æ af 34$ë,/')#ë6%#4/2Ý7)$4(Èú $/7.4/ ùÞÅ ÈÈhjna gh]jYf\
      Zëa    Æ af 34$ë,/')#ë6%#4/2Ý7)$4(Èú $/7.4/ ùÞÅ ÈÈ\jm_a gh]jYf\
      ghëa   Æ af 34$ë,/')#ë6%#4/2Ýý $/7.4/ ùÞÅ ÈÈhgjl rY arZgj gh]jY[ab]
      j]këg  Æ gml 34$ë,/')#ë6%#4/2Ý7)$4(Èú $/7.4/ ùÞÅ ÈÈj]rmdlYl
      r]jgëg Æ gml 34$ë,/')#Å ÈÈ ka_fYd \Y b] j]rmdlYl fmdY
      g^ëg   Æ gml 34$ë,/')#ÞÅ ÈÈ ka_fYd \Y b] \gkdg \g hj]cgjY[]fbY ghk]_Y
%.$ !,5Å

!2#()4%#452% Z]`YnagjYd /& !,5 )3

   [gfklYfl  dû7)$4( Æ fYlmjYd Æđ afl]_]jÝ[]adÝdg_ûÝj]YdÝ7)$4(ÞÞÞÅ
   ka_fYd    Y\\ëj]kÄ kmZëj]kÄ gjëj]kÄ Yf\ëj]kÄj]këkÆ  34$ë,/')#ë6%#4/2Ý7)$4(Èú
$/7.4/ ùÞÅ


"%'af

   ÈÈ Y\\alagf
   Y\\ëj]k Ēđ kl\ëdg_a[ën][lgjÝmfka_f]\ÝYëaÞ č mfka_f]\ÝZëaÞÞÅ
   ÈÈ kmZljY[lagf
   kmZëj]k Ēđ kl\ëdg_a[ën][lgjÝmfka_f]\ÝYëaÞ È mfka_f]\ÝZëaÞÞÅ
   ÈÈ Yf\ _Yl]
   Yf\ëj]k Ēđ Yëa Yf\ ZëaÅ
   ÈÈ gj _Yl]
   gjëj]k Ēđ Yëa gj ZëaÅ


   ÈÈ 3%,%#4 2%35,4
   j]këg Ēđ j]këkÅ
```

```
   oal` ghëa k]d][l
      j]këk Ēđ Yf\ëj]k o`]f Yf\ëghÄ ÈÈYf\
               gjëj]k  o`]f gjëghÄ ÈÈgj
               Y\\ëj]k o`]f Y\\ëghÄ ÈÈY\\
               kmZëj]k o`]f kmZëghÄ ÈÈkmZ
               Ýgl`]jk đē ìúìÞ o`]f gl`]jkÅ


   ÈÈ 0gklYndbY r]jgëg fY ú mcgdacg b] j]rmdlYl gh]jY[ab] ù
   r]jgëg Ēđ ìúì o`]f j]këk đ kl\ëdg_a[ën][lgjÝlgëmfka_f]\ÝùÄ7)$4(ÞÞ ]dk]
             ìùìÅ

   ÈÈ 0j]cgjY[]fb] k] \]kYnY cY\Y mdYra aeYbm akla rfYcÄ Y ardYr jYrda[al
   g^ëg Ēđ ìúì o`]f ÝÝghëađíùùùúúí Yf\ ÝYëaÝ7)$4(ÈúÞđZëaÝ7)$4(ÈúÞÞ Yf\
ÝÝYëaÝ7)$4(ÈúÞ pgj j]këkÝ7)$4(ÈúÞÞđìúìÞÞ gj Ýghëađíúùùúúí Yf\
ÝYëaÝ7)$4(ÈúÞđj]këkÝ7)$4(ÈúÞÞ Yf\ ÝÝYëaÝ7)$4(ÈúÞ pgj ZëaÝ7)$4(ÈúÞÞđìúìÞÞÞ ]dk]
           ìùìÅ


%.$ Z]`YnagjYdÅ
```

3.4 DATAPATH

Sledeći kodni listing predstavlja vhdl implementaciju ***datapath*** celine. Unutar DATAPATH modula (slika 22) instancirane su prethodno opisane komponente (ALU, REGISTER_BANK i IMMEDIATE), implementirana je dodatna logika koja je neophonda (sabirači, multiplekseri, komparatori, idt.) i sve to je povezano na način koji je objašnjen u delovima 2.1 do 2.1.5.

```
daZjYjq a]]]Å
mk] a]]]Çkl\ëdg_a[ëúúÿýÇYddÅ
mk] a]]]Çfme]ja[ëkl\ÇYddÅ


]flalq \YlYëhYl` ak
   _]f]ja[ Ý$!4!ë7)$4( Æ hgkalan] Æđ üûÞÅ
   hgjlÝ
      ÈÈ ããäãäãäãäã 'dgZYdfY kaf`jgfarY[abY ããäãäãäãäãäãäãäãäãäãäã
      [dc                Æ af  kl\ëdg_a[Å
      j]k]l              Æ af  kl\ëdg_a[Å
      ÈÈ ããäãäãäãäã )fl]j^]bk cY -]egjaba rY afkljmc[ab] ããäãäã
      afkljëe]eëY\\j]kkëg Æ gml kl\ëdg_a[ën][lgjÝüú \goflg ùÞÅ
      afkljëe]eëj]Y\ëa    Æ af  kl\ëdg_a[ën][lgjÝüú \goflg ùÞÅ
      afkljm[lagfëg       Æ gml kl\ëdg_a[ën][lgjÝüú \goflg ùÞÅ
      ÈÈ ããäãäãäãäã )fl]j^]bk cY -]egjaba rY hg\Ylc] ããäãäã
      \YlYëe]eëY\\j]kkëg  Æ gml kl\ëdg_a[ën][lgjÝüú \goflg ùÞÅ
      \YlYëe]eëojal]ëg    Æ gml kl\ëdg_a[ën][lgjÝüú \goflg ùÞÅ
      \YlYëe]eëj]Y\ëa     Æ af  kl\ëdg_a[ën][lgjÝüú \goflg ùÞÅ
      ÈÈ ããäãäãäãäã +gfljgdfa ka_fYda ããäãäãäãäãäãäãäãäãäãäãäãäã
      e]eëlgëj]_ëa        Æ af  kl\ëdg_a[Å
      Ydmëghëa            Æ af  kl\ëdg_a[ën][lgjÝý \goflg ùÞÅ
      h[ëf]plëk]dëa       Æ af  kl\ëdg_a[Å
      Ydmëkj[ëa           Æ af  kl\ëdg_a[Å
      j\ëo]ëa             Æ af  kl\ëdg_a[Å
      ÈÈ ããäãäãäãäã 3lYlmkfa ka_fYda ããäãäãäãäãäãäãäãäãäãäãäãäã
      ZjYf[`ë[gf\alagfëg  Æ gml kl\ëdg_a[
    ÈÈ ããäãäãäãäãäãäãäãäãäãäãäãäãäãäãäãäãäãäãäãäãäãäãäãäãäãäãäãäã
      ÞÅ

]f\ ]flalqÅ


Yj[`al][lmj] "]`YnagjYd g^ \YlYëhYl` ak
   ÈÈãäãäãäãäãäãäãäã2%')342)ãäãäãäãäãäãäãäãäãäãäãäãäãäãäãäãäã
   ka_fYd h[ëj]_ëkÄ h[ëf]plëk                    Æ kl\ëdg_a[ën][lgj Ýüú \goflg ùÞÅ
   ÈÈãäãäãäãäãäãäãäãäãäãäãäãäãäãäãäãäãäãäãäãäãäãäãäãäãäãäãäãäãäãäã
   ÈÈãäãäãäãäãäãäã3)'.!,)ãäãäãäãäãäãäãäãäãäãäãäãäãäãäãäãäãäãäã
   ka_fYd afkljm[lagfëk                          Æ kl\ëdg_a[ën][lgjÝüú \goflg ùÞÅ
```

```
   ka_fYd h[ëY\\]jëk                              Æ kl\ëdg_a[ën][lgjÝüú \goflg ùÞÅ
   ka_fYd ZjYf[`ëY\\]jëk                          Æ kl\ëdg_a[ën][lgjÝüú \goflg ùÞÅ
   ka_fYd jkúë\YlYëkÄ jkûë\YlYëkÄ j\ë\YlYëk     Æ kl\ëdg_a[ën][lgj Ýüú \goflg ùÞÅ
   ka_fYd aee]\aYl]ë]pl]f\]\ëkÄ ]pl]f\]\ë\YlYëk Æ kl\ëdg_a[ën][lgjÝüú \goflg ùÞÅ
   ÈÈ !d5 ka_fYda
   ka_fYd Ydmër]jgëkÄ Ydmëg^ëgëk                Æ kl\ëdg_a[Å
   ka_fYd ZëkÄ Yëk                                Æ kl\ëdg_a[ën][lgjÝüú \goflg ùÞÅ
   ka_fYd Ydmëj]kmdlëk                            Æ kl\ëdg_a[ën][lgjÝüú \goflg ùÞÅ
   ÈÈ3a_fYda _jYfYfbY Ý]f_Ç ZjYf[` ka_fYdkÞÇ
   ka_fYd Z[[                                     Æ kl\ëdg_a[Å
ÈÈãããããããããããããããããããããããããããããããããããããããããããããããããããããããããã
Z]_af

   ÈÈãããããããããã3]cn]f[abYdfY dg_acYãããããããããããããããããããã
   h[ëhjg[ Æ hjg[]kk Ý[dcÞ ak
   Z]_af
      a^ Ýjakaf_ë]\_]Ý[dcÞÞ l`]f
         a^ Ýj]k]l đ ìùìÞl`]f
            h[ëj]_ëk Ēđ Ýgl`]jk đē ìùìÞÅ
         ]dk]
            h[ëj]_ëk Ēđ h[ëf]plëkÅ
         ]f\ a^Å
      ]f\ a^Å
   ]f\ hjg[]kkÅ
   ÈÈããããããããããããããããããããããããããããããããããããããããããããããããããããããããã

   ÈÈãããããããããã+geZafY[agfY dg_acYãããããããããããããããããããã
   Z[[ Ēđ afkljm[lagfëkÝúûÞÅ

   ÈÈ kYZajY[ rY mn][YnYfb] hjg_jYekcg_ ZjgbY[Y Ýkd]\][Y afkljmc[abYÞ
   h[ëY\\]jëk     Ēđ kl\ëdg_a[ën][lgjÝmfka_f]\Ýh[ëj]_ëkÞ č lgëmfka_f]\ÝýÄ
$!4!ë7)$4(ÞÞÅ
   ÈÈ kYZajY[ rY mkdgnf] kcgcgn]
   ZjYf[`ëY\\]jëk Ēđ kl\ëdg_a[ën][lgjÝmfka_f]\Ýaee]\aYl]ë]pl]f\]\ëkÞ č
mfka_f]\Ýh[ëj]_ëkÞÞÅ

   ÈÈ 0jgn]jY mkdgnY kcgcY
   ZjYf[`ë[gf\alagfëg Ēđ ìúì o`]f Yëk đ Zëk ]dk]
                         ìùìÅ

   ÈÈ -58 cgba g\j]\bmb] kd]\][m nj]\fgkl rY hjg_jYekca ZjgbY[Ç
   ÈÈ !cg k] f] \]ka kcgc hjg_jYekca ZjgbY[ k] mn][YnY rY ýÇ
   oal` h[ëf]plëk]dëa k]d][l
      h[ëf]plëk Ēđ h[ëY\\]jëk o`]f ìùìÄ
      ZjYf[`ëY\\]jëk          o`]f gl`]jkÅ

   ÈÈ -58 cgba g\j]\bmb] kd]\][m nj]\fgkl rY Z mdYr !,5 b]\afa[]Ç
```

```
 Zëk Ēđ jkûë\YlYëk o`]f Ydmëkj[ëa đ ìùì ]dk]
        aee]\aYl]ë]pl]f\]\ëkÅ
 ÈÈ !rmjajYfb] Y mdYrY !,5 b]\afa[]
 Yëk Ēđ jkúë\YlYëkÅ

ÈÈ-58 cgba g\j]\bmb] klY k] mhakmb] m g\j]\akfa j]_aklYjÝj\ë\YlYëkÞ
 j\ë\YlYëk Ēđ \YlYëe]eëj]Y\ëa o`]f e]eëlgëj]_ëa đ ìúì ]dk]
              Ydmëj]kmdlëkÅ
 ÈÈããããããããããããããããããããããããããããããããããããããããããããããããããã

 ÈÈããããããããããã)fklYf[ajYfbYããããããããããããããããããããããããããã

 ÈÈ2]_aklYjkcY ZYfcY
 j]_akl]jëZYfcëú Æ ]flalq ogjcÇj]_akl]jëZYfc
    _]f]ja[ eYh Ý
       7)$4( đē üûÞ
    hgjl eYh Ý
       [dc           đē [dcÄ
       j]k]l         đē j]k]lÄ
       j\ëo]ëa       đē j\ëo]ëaÄ
       jkúëY\\j]kkëa đē afkljm[lagfëk ÝúĂ \goflg úþÞÄ
       jkûëY\\j]kkëa đē afkljm[lagfëk Ýûý \goflg ûùÞÄ
       jkúë\YlYëg    đē jkúë\YlYëkÄ
       jkûë\YlYëg    đē jkûë\YlYëkÄ
       j\ëY\\j]kkëa  đē afkljm[lagfëk Ýúú \goflg ĀÞÄ
       j\ë\YlYëa     đē j\ë\YlYëkÞÅ


 ÈÈ -g\md rY hjgkaj]fb] aee]\aYl] hgdbY afkljmc[ab]
 aee]\aYl]ëú Æ ]flalq ogjcÇaee]\aYl]
    hgjl eYh Ý
       afkljm[lagfëa        đē afkljm[lagfëkÄ
       aee]\aYl]ë]pl]f\]\ëg đē aee]\aYl]ë]pl]f\]\ëk
       ÞÅ

 ÈÈ !jale]la[cg dg_a[cY b]\afa[Y
 !,5ëú Æ ]flalq ogjcÇ!,5
    _]f]ja[ eYh Ý
       7)$4( đē $!4!ë7)$4(Þ
    hgjl eYh Ý
       Yëa    đē YëkÄ
       Zëa    đē ZëkÄ
       ghëa   đē YdmëghëaÄ
       j]këg  đē Ydmëj]kmdlëkÄ
       r]jgëg đē Ydmër]jgëkÄ
       g^ëg   đē Ydmëg^ëgëkÞÅ
```

```
   ÈÈããããããããããããããããããããããããããããããããããããããããããããããããããããããããããããããã

   ÈÈããããããããããã5dYra×)rdYraããããããããããããããããããããããããããããããããããããããã
   ÈÈ +Y [gfljgdhYl`Èm
   afkljm[lagfëg       Ēđ afkljm[lagfëkÅ
   ÈÈ 3Y e]egjabge rY afkljmc[ab]
   afkljm[lagfëk       Ēđ afkljëe]eëj]Y\ëaÅ
   ÈÈ 3Y e]egjabge rY hg\Ylc]
   afkljëe]eëY\\j]kkëg Ēđ h[ëj]_ëkÅ
   \YlYëe]eëY\\j]kkëg  Ēđ Ydmëj]kmdlëkÅ
   \YlYëe]eëojal]ëg    Ēđ jkûë\YlYëkÅ
]f\ Yj[`al][lmj]Å
```

3.5 CTRL_DECODER

Sledeći kodni listing predstavlja vhdl implementaciju ***ctrl_decoder*** modula. Njegova funkcionalnost opisana je u poglavlju 2.2 ovih vežbi.

```
daZjYjq a]]]Å
mk] a]]]Çkl\ëdg_a[ëúúÿýÇYddÅ
mk] a]]]Çfme]ja[ëkl\ÇYddÅ

]flalq [ljdë\][g\]j ak
   hgjl Ý
      ÈÈããããããããããã /h[g\] hgdb] afkljmc[ab]ããããããããããã
      gh[g\]ëa      Æ af  kl\ëdg_a[ën][lgj Ýÿ \goflg ùÞÅ
      ÈÈããããããããããã +gfljgdfa ka_fYdaãããããããããããããããã
      ZjYf[`ëg      Æ gml kl\ëdg_a[Å
      e]eëlgëj]_ëg  Æ gml kl\ëdg_a[Å
      \YlYëe]eëo]ëg Æ gml kl\ëdg_a[Å
      Ydmëkj[ëg     Æ gml kl\ëdg_a[Å
      j\ëo]ëg       Æ gml kl\ëdg_a[Å
      YdmëûZalëghëg Æ gml kl\ëdg_a[ën][lgjÝú \goflg ùÞ
      ÞÅ
]f\ ]flalqÅ

Yj[`al][lmj] Z]`YnagjYd g^ [ljdë\][g\]j ak
Z]_af

   [gflgdë\][ Æ hjg[]kkÝgh[g\]ëaÞak
   Z]_af
      ÈÈããã0g\jYrme]nYf] nj]\fgklããã
      ZjYf[`ëg      Ēđ ìùìÅ
      e]eëlgëj]_ëg  Ēđ ìùìÅ
      \YlYëe]eëo]ëg Ēđ ìùìÅ
      Ydmëkj[ëg     Ēđ ìùìÅ
```

```
      j\ëo]ëg       Ēđ ìùìÅ
      YdmëûZalëghëg Ēđ íùùíÅ
      ÈÈãããããããããããããããããããããããããã
      [Yk] gh[g\]ëaÝÿ \goflg ûÞ ak
         o`]f íùùùùùí đē               ÈÈ,/!$Ä þn è ^mf[lü
            YdmëûZalëghëg Ēđ íùùíÅ
            e]eëlgëj]_ëg  Ēđ ìúìÅ
            Ydmëkj[ëg     Ēđ ìúìÅ
            j\ëo]ëg       Ēđ ìúìÅ
         o`]f íùúùùùí đē               ÈÈ34/2%Ä ün è ^mf[lü
            YdmëûZalëghëg Ēđ íùùíÅ
            \YlYëe]eëo]ëg Ēđ ìúìÅ
            Ydmëkj[ëg     Ēđ ìúìÅ
         o`]f íùúúùùí đē               ÈÈ2 lqh]Ä
            YdmëûZalëghëg Ēđ íúùíÅ
            j\ëo]ëg       Ēđ ìúìÅ
         o`]f íùùúùùí đē               ÈÈ) lqh]
            YdmëûZalëghëg Ēđ íúúíÅ
            Ydmëkj[ëg     Ēđ ìúìÅ
            j\ëo]ëg       Ēđ ìúìÅ
         o`]f íúúùùùí đē               ÈÈ" lqh] "%1
            YdmëûZalëghëg Ēđ íùúíÅ
            ZjYf[`ëg      Ēđ ìúìÅ
         o`]f gl`]jk đē
      ]f\ [Yk]Å
   ]f\ hjg[]kkÅ

]f\ Yj[`al][lmj]Å
```

3.5 ALU_DECODER

Sledeći kodni listing predstavlja vhdl implementaciju ***alu_decoder*** modula. Njegova funkcionalnost opisana je u poglavlju 2.2 ovih vežbi.

```
daZjYjq a]]]Å
mk] a]]]Çkl\ëdg_a[ëúúÿýÇYddÅ
mk] a]]]Çfme]ja[ëkl\ÇYddÅ
mk] ogjcÇYdmëghkëhc_ÇYddÅ

]flalq Ydmë\][g\]j ak
   hgjl Ý
      ÈÈãããããããã #gfljgdhYl` mdYra ããããããããã
      YdmëûZalëghëa Æ af  kl\ëdg_a[ën][lgjÝú \goflg ùÞÅ
      ÈÈãããããããã 0gdbY afkljmc[ab] ããããããã
      ^mf[lüëa      Æ af  kl\ëdg_a[ën][lgj Ýû \goflg ùÞÅ
      ^mf[lÃëa      Æ af  kl\ëdg_a[ën][lgj Ýÿ \goflg ùÞÅ
      ÈÈãããããããã $YlYhYl` ardYra ãããããããã
```

```
      Ydmëghëg      Æ gml kl\ëdg_a[ën][lgjÝý \goflg ùÞÞÅ
]f\ ]flalqÅ

Yj[`al][lmj] Z]`YnagjYd g^ Ydmë\][g\]j ak
Z]_af

   ÈÈ+geZafY[agfY dg_acY cgbY fY gkfgnm af^gjeY[ab] ar [ljdë\][g\]j
   ÈÈeg\mdY hgklYndbY Ydmëghëg fY g\j]\b]fm nj]\fgklÄ hja []em lY
   ÈÈnj]\fgkl hj]\klYndbY r]db]fm gh]jY[abmÇ
   Ydmë\][ Æ hjg[]kkÝYdmëûZalëghëaÄ ^mf[lüëaÄ ^mf[lÃëaÞak
   Z]_af
      Ydmëghëg Ēđ íùùùùùíÅ              ÈÈ0g\jYrme]nYfY nj]\fgkl

      [Yk] YdmëûZalëghëa ak
         o`]f íùùí đē
            Ydmëghëg Ēđ Y\\ëghÅ
         o`]f íùúí đē
            Ydmëghëg Ēđ ]iëghÅ
         o`]f gl`]jk đē
            [Yk] ^mf[lüëa ak
               o`]f íùùùí đē
                  Ydmëghëg Ēđ Y\\ëghÅ
                  a^Ý^mf[lÃëaÝþÞ đ ìúìÞl`]f
                     Ydmëghëg Ēđ kmZëghÅ
                  ]f\ a^Å
               o`]f íúúùí đē
                  Ydmëghëg Ēđ gjëghÅ
               o`]f gl`]jk đē
                  Ydmëghëg Ēđ Yf\ëghÅ
            ]f\ [Yk]Å
      ]f\ [Yk]Å
   ]f\ hjg[]kkÅ

]f\ Yj[`al][lmj]Å
```

3.5 CONTROL_PATH

Unutar control_path celine su instancirani CTRL_DECODER i ALU_DECODER i izvršeno je njihovo povezivanje na način objašnjen u 2.2. Sledeći kodni listing predstavlja vhdl implementacija control_path celine:

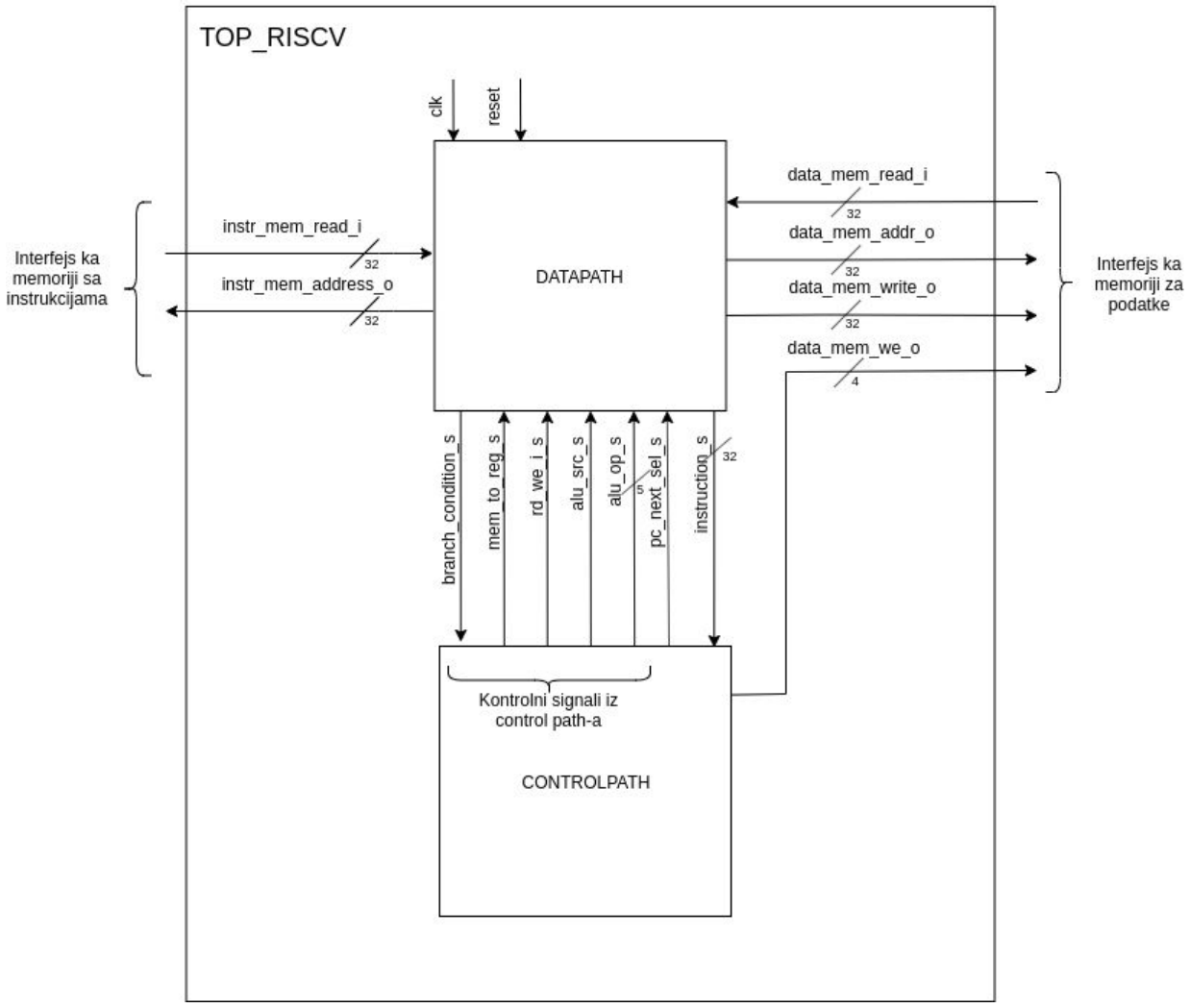
```
daZjYjq a]]]Å
mk] a]]]Çkl\ëdg_a[ëúúÿýÇYddÅ
mk] a]]]Çfme]ja[ëkl\ÇYddÅ

]flalq [gfljgdëhYl` ak
   hgjl Ý[dc               Æ af  kl\ëdg_a[Å
        j]k]l              Æ af  kl\ëdg_a[Å
```

```
         ÈÈ ããããããããã )fl]j^]bk rY hja`nYl afkljmc[ab] ar \YlYhYl`ÈYãããããããã
         afkljm[lagfëa      Æ af  kl\ëdg_a[ën][lgj Ýüú \goflg ùÞÅ
         ÈÈ ããããããããã +gfljgdfa afl]^]bk ãããããããããããããããããããããããããããããããã
         e]eëlgëj]_ëg       Æ gml kl\ëdg_a[Å
         Ydmëghëg           Æ gml kl\ëdg_a[ën][lgjÝý \goflg ùÞÅ
         h[ëf]plëk]dëg      Æ gml  kl\ëdg_a[Å
         Ydmëkj[ëg          Æ gml kl\ëdg_a[Å
         j\ëo]ëg            Æ gml kl\ëdg_a[Å
         ÈÈããããããããã 5dYrfa 3lYlmkfa afl]j^]bk
ããããããããããããããããããããããããããããããããããããããããããã
         ZjYf[`ë[gf\alagfëa Æ af  kl\ëdg_a[Å
         ÈÈããããããããã )rdYrfa 3lYlmkfa afl]j^]bk
ããããããããããããããããããããããããããããããããããããããããããã
         \YlYëe]eëo]ëg      Æ gml kl\ëdg_a[ën][lgjÝü \goflg ùÞ
         ÞÅ
]f\ ]flalqÅ

Yj[`al][lmj] Z]`YnagjYd g^ [gfljgdëhYl` ak
   ka_fYd YdmëûZalëghëk Æ kl\ëdg_a[ën][lgjÝú \goflg ùÞÅ
   ka_fYd \YlYëe]eëo]ëk Æ kl\ëdg_a[Å
   ka_fYd ZjYf[`ëkÆ kl\ëdg_a[Å
Z]_af

   hjg[]kk ÝZjYf[`ë[gf\alagfëaÄ ZjYf[`ëkÞak
   Z]_af
      h[ëf]plëk]dëg Ēđ ìùìÅ
      a^ ÝZjYf[`ëk đ ìúì Yf\ ZjYf[`ë[gf\alagfëa đ ìúìÞl`]f
         h[ëf]plëk]dëg Ēđ ìúìÅ
      ]f\ a^Å
   ]f\ hjg[]kkÅ

   [ljdë\][ Æ ]flalq ogjcÇ[ljdë\][g\]jÝZ]`YnagjYdÞ
      hgjl eYhÝ
         gh[g\]ëa      đē afkljm[lagfëaÝÿ \goflg ùÞÄ
         ZjYf[`ëg      đē ZjYf[`ëkÄ
         e]eëlgëj]_ëg  đē e]eëlgëj]_ëgÄ
         \YlYëe]eëo]ëg đē \YlYëe]eëo]ëkÄ
         Ydmëkj[ëg     đē Ydmëkj[ëgÄ
         j\ëo]ëg       đē j\ëo]ëgÄ
         YdmëûZalëghëg đē YdmëûZalëghëkÞÅ

   Ydmë\][ Æ ]flalq ogjcÇYdmë\][g\]jÝZ]`YnagjYdÞ
      hgjl eYhÝ
         YdmëûZalëghëa đē YdmëûZalëghëkÄ
         ^mf[lüëa      đē afkljm[lagfëaÝúý \goflg úûÞÄ
         ^mf[lĀëa      đē afkljm[lagfëaÝüú \goflg ûþÞÄ
         Ydmëghëg      đē YdmëghëgÞÅ
```

```
   ÈÈãããããããããããããã)rdYraããããããããããããããããããããããã
   \YlYëe]eëo]ëg Ēđ Ýgl`]jk đē Ý\YlYëe]eëo]ëkÞÞÅ


]f\ Yj[`al][lmj]Å
```

3.6 TOP_RISCV

Na samom kraju neophodno je instancirati i povezati DATA_PATH i CONTROL_PATH i to je urađeno na sledeći način:



Slika 23. TOP_RISCV

Kao što je prethodno pomenuto interfejs procesora je sačinjen od portova za komunikaciju sa memorijom za podatke i portova za komunikaciju sa memorijom za instrukcije.

Sa memorijom za instrukcije se komunicira preko dva porta:

- ***instr_mem_read_i*** porta preko koga se primaju instrukcije
- instr_mem_addr_o porta preko koga se šalje adresa naredne instrukcije

Komunikacija sa memorijom za podatke se obavlja preko 4 porta:

- ***data_mem_read_i*** ulazni port preko koga se prima podataka iz memorije
- data_mem_addr_o izlazni port preko koga se šalje adresa sa koje podatak čita ili na koju se podatak upisuje.
- data_mem_write_o izlazni port preko koga se šalje vrednost koju treba upisati u memoriju.
- data_mem_we izlazni port preko koga se šalje signal dozvolje upisa.

Ako se pogleda slika 23 može se primetiti da je port dozvole upisa data_mem_we_o četvorobitan i razlog za to je memorija sa kojom se komunicira. Ona je napravljena tako da se u nju odjednom može upisati 1, 2, 3 ili 4 bajta u zavisnosti od vrednosti tog porta (informacija koja se upisuje nalazi se na data_mem_write_o portu). Sledeći primer ilustruje kako vrednost tog porta određuje koliko bajtova se upisuje:

data_mem_we_o = 1111_2 => upisuju se 4 bajta
data_mem_we_o = 1110_2 => upisuju se gornja 3 bajta
data_mem_we_o = 0110_2 => upisuju se središnja 2 bajta bajta

U slučaju ovog procesora data_mem_we_o može da ima dve vrednosti: 0000_2 ili 1111_2. To je zato što za sada samo jedna instrukcija vrši upis u procesor (***sw***) i ona vrši upis informacije od 32 bita. Ukoliko se čitalac odluči da proširi set instrukcija koji procesor podržava i reši da implementira ***sh (store halfword)*** instrukciju, onda će prilikom izvršavanja ove instrukcije morati da na data_mem_we_o port postavi vrednost 0011_2.

Sledeći kodni listing predstavlja vhdl implementaciju TOP_RISCV celine:

```
daZjYjq a]]]Å
mk] a]]]Çkl\ëdg_a[ëúúÿýÇYddÅ
mk] a]]]Çfme]ja[ëkl\ÇYddÅ

]flalq 4/0ë2)3#6 ak
   _]f]ja[ Ý$!4!ë7)$4( Æ hgkalan] Æđ üûÞÅ
   hgjlÝ
      ÈÈ ããäãäãäãã 'dgZYdfY kaf`jgfarY[abY ããäãäãäãäãäãäãäãäãäãäã
      [dc                Æ af  kl\ëdg_a[Å
```

```
      j]k]l               Æ af  kl\ëdg_a[Å
      ÈÈ ããããããããã )fl]j^]bk cY -]egjaba rY afkljmc[ab] ããããã
      afkljëe]eëY\\j]kkëg Æ gml kl\ëdg_a[ën][lgj Ýüú \goflg ùÞÅ
      afkljëe]eëj]Y\ëa    Æ af  kl\ëdg_a[ën][lgjÝüú \goflg ùÞÅ
      ÈÈ ããããããããã )fl]j^]bk cY -]egjaba rY hg\Ylc] ããããããããã
      \YlYëe]eëo]ëg       Æ gml kl\ëdg_a[ën][lgjÝü \goflg ùÞÅ
      \YlYëe]eëY\\j]kkëg  Æ gml kl\ëdg_a[ën][lgjÝüú \goflg ùÞÅ
      \YlYëe]eëojal]ëg    Æ gml kl\ëdg_a[ën][lgjÝüú \goflg ùÞÅ
      \YlYëe]eëj]Y\ëa     Æ af  kl\ëdg_a[ën][lgj Ýüú \goflg ùÞÞÅ
]f\ ]flalqÅ

Yj[`al][lmj] kljm[lmjYd g^ 4/0ë2)3#6 ak
   ka_fYd afkljm[lagfëk      Æ kl\ëdg_a[ën][lgjÝüú \goflg ùÞÅ
   ka_fYd e]eëlgëj]_ëk       Æ kl\ëdg_a[Å
   ka_fYd Ydmëghëk           Æ kl\ëdg_a[ën][lgj Ýý \goflg ùÞÅ
   ka_fYd Ydmëkj[ëk          Æ kl\ëdg_a[Å
   ka_fYd j\ëo]ëk            Æ kl\ëdg_a[Å
   ka_fYd ZjYf[`ë[gf\alagfëk Æ kl\ëdg_a[Å
   ka_fYd h[ëf]plëk]dëk      Æ kl\ëdg_a[Å
Z]_af

   \YlYëhYl`ëú Æ ]flalq ogjcÇ\YlYëhYl`
      _]f]ja[ eYh Ý
         $!4!ë7)$4( đē $!4!ë7)$4(Þ
      hgjl eYh Ý
         [dc                đē [dcÄ
         j]k]l              đē j]k]lÄ
         afkljëe]eëY\\j]kkëg đē afkljëe]eëY\\j]kkëgÄ
         afkljëe]eëj]Y\ëa   đē afkljëe]eëj]Y\ëaÄ
         afkljm[lagfëg      đē afkljm[lagfëkÄ
         \YlYëe]eëY\\j]kkëg đē \YlYëe]eëY\\j]kkëgÄ
         \YlYëe]eëojal]ëg   đē \YlYëe]eëojal]ëgÄ
         \YlYëe]eëj]Y\ëa    đē \YlYëe]eëj]Y\ëaÄ
         e]eëlgëj]_ëa       đē e]eëlgëj]_ëkÄ
         Ydmëghëa           đē YdmëghëkÄ
         h[ëf]plëk]dëa      đē h[ëf]plëk]dëkÄ
         Ydmëkj[ëa          đē Ydmëkj[ëkÄ
         j\ëo]ëa            đē j\ëo]ëkÄ
         ZjYf[`ë[gf\alagfëg đē ZjYf[`ë[gf\alagfëkÞÅ

   [gfljgdëhYl`ëú Æ ]flalq ogjcÇ[gfljgdëhYl`
      hgjl eYh Ý
         [dc               đē [dcÄ
         j]k]l             đē j]k]lÄ
         afkljm[lagfëa     đē afkljm[lagfëkÄ
         e]eëlgëj]_ëg      đē e]eëlgëj]_ëkÄ
         Ydmëghëg          đē YdmëghëkÄ
```

```
         h[ëf]plëk]dëg      đē h[ëf]plëk]dëkÄ
         Ydmëkj[ëg          đē Ydmëkj[ëkÄ
         j\ëo]ëg            đē j\ëo]ëkÄ
         ZjYf[`ë[gf\alagfëa đē ZjYf[`ë[gf\alagfëkÄ
         \YlYëe]eëo]ëg      đē \YlYëe]eëo]ëgÞÅ
   ÈÈãããããããããããããããããããããããããããããããããã


]f\ Yj[`al][lmj]Å
```

3.7 Paketi

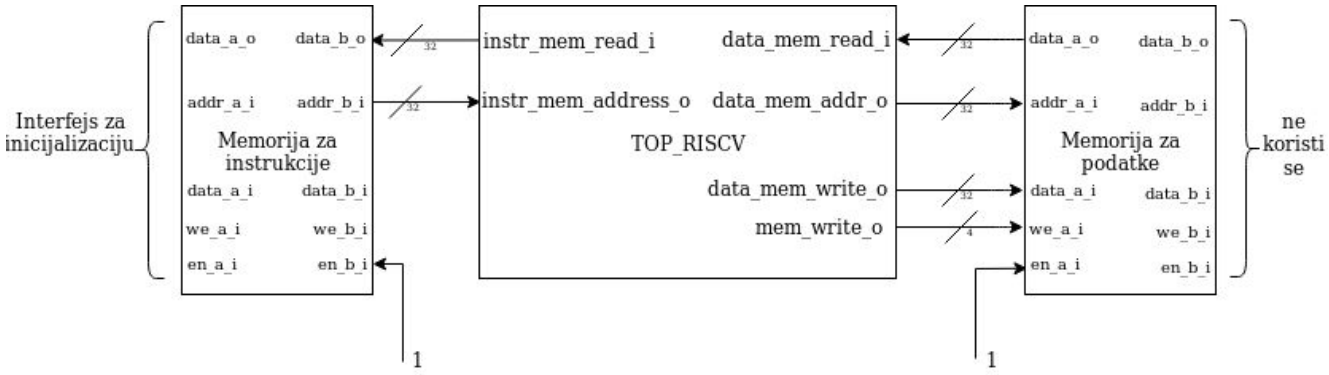
Da bi se olakšala čitljivost koda napravljen je alu_ops_pkg.vhd fajl u kome su smeštene konstante korišćene u nekim od fajlova. One predstavljaju sve operacije koje aritmetičko logička jedinica može da koristi. Takođe za potrebe pisanja verifikacionog okruženja neohpodno je uključiti i txt_util.vhd paket u kome su implementirane funkcije za rad sa tekstualnim fajlovima. On se nalazi u priloženim fajlovima. Sledeći kodni listing predstavlja ***alu_ops_pkg.vhd*** paket:

```
daZjYjq a]]]Å
mk] a]]]Çkl\ëdg_a[ëúúÿýÇYddÅ

hY[cY_] Ydmëghkëhc_ ak

   ÈÈ !,5 /0 #/$%
   [gfklYfl Yf\ëgh    Æ kl\ëdg_a[ën][lgj Ýý \goflg ùÞ Æđ íùùùùùíÅ  ÈÈÈē Zaloak] Yf\
   [gfklYfl gjëgh     Æ kl\ëdg_a[ën][lgj Ýý \goflg ùÞ Æđ íùùùùúíÅ  ÈÈÈē Zaloak] gj
   [gfklYfl pgjëgh    Æ kl\ëdg_a[ën][lgj Ýý \goflg ùÞ Æđ íùùùúúíÅ  ÈÈÈē Zaloak] pgj
   [gfklYfl Y\\ëgh    Æ kl\ëdg_a[ën][lgj Ýý \goflg ùÞ Æđ íùùùúùíÅ  ÈÈÈē Y\\ Yëa Yf\
Zëa
   [gfklYfl kmZëgh    Æ kl\ëdg_a[ën][lgj Ýý \goflg ùÞ Æđ íùùúúùíÅ  ÈÈÈē kmZ Yëa Yf\
Zëa
   [gfklYfl ]iëgh     Æ kl\ëdg_a[ën][lgj Ýý \goflg ùÞ Æđ íúùúúúíÅ  ÈÈÈē  k]l ]imYd
   [gfklYfl dlkëgh    Æ kl\ëdg_a[ën][lgj Ýý \goflg ùÞ Æđ íúùúùùíÅ  ÈÈÈē k]l d]kk
l`Yf ka_f]\
   [gfklYfl dlmëgh    Æ kl\ëdg_a[ën][lgj Ýý \goflg ùÞ Æđ íúùúùúíÅ  ÈÈÈē k]l d]kk
l`Yf mfka_f]\
   [gfklYfl kddëgh    Æ kl\ëdg_a[ën][lgj Ýý \goflg ùÞ Æđ íúùúúùíÅ  ÈÈÈē k`a^l d]^l
dg_a[
   [gfklYfl kjdëgh    Æ kl\ëdg_a[ën][lgj Ýý \goflg ùÞ Æđ íúùúúúíÅ  ÈÈÈē k`a^l ja_`l
dg_a[
   [gfklYfl kjYëgh    Æ kl\ëdg_a[ën][lgj Ýý \goflg ùÞ Æđ íúúùùùíÅ  ÈÈÈē k`a^l ja_`l
Yjal`e]la[
   [gfklYfl emdmëgh   Æ kl\ëdg_a[ën][lgj Ýý \goflg ùÞ Æđ íúúùùúíÅ  ÈÈÈē emdlahdq
dgo]j
   [gfklYfl emd`këgh  Æ kl\ëdg_a[ën][lgj Ýý \goflg ùÞ Æđ íúúúúùíÅ  ÈÈÈē emdlahdq
`a_`]j ka_f]\
```

```
   [gfklYfl emd`kmëgh Æ kl\ëdg_a[ën][lgj Ýý \goflg ùÞ Æđ íùúùúúíÅ  ÈÈÈē emdlahdq
`a_`]j ka_f]\ Yf\ mfka_f]\
   [gfklYfl emd`mëgh  Æ kl\ëdg_a[ën][lgj Ýý \goflg ùÞ Æđ íùúúùùíÅ  ÈÈÈē emdlahdq
`a_`]j mfka_f]\
   [gfklYfl \anmëgh   Æ kl\ëdg_a[ën][lgj Ýý \goflg ùÞ Æđ íùúúùùíÅ  ÈÈÈē \ana\]
mfka_f]\
   [gfklYfl \ankëgh   Æ kl\ëdg_a[ën][lgj Ýý \goflg ùÞ Æđ íùúúùùíÅ  ÈÈÈē \ana\]
ka_f]\
   [gfklYfl j]emëgh   Æ kl\ëdg_a[ën][lgj Ýý \goflg ùÞ Æđ íùúúùùíÅ  ÈÈÈē j]eaf\]j
mfka_f]\
   [gfklYfl j]ekëgh   Æ kl\ëdg_a[ën][lgj Ýý \goflg ùÞ Æđ íúùùùùíÅ  ÈÈÈē j]eaf\]j
ka_f]\


]f\ hY[cY_] Ydmëghkëhc_Å
```

3.8 Verifikaciono okruženje.

Da bi se testirao procesor napisan je jednostavan ***testbench.*** Unutar njega je instanciran TOP_RISCV modul i dve memorije koje predstavljaju memoriju sa instrukcijama i memoriju sa podacima. Slika 24 to ilustruje:



Slika 24. Verifikaciono okruženje

Testiranja ispravnog rada procesore izvršeno je tako što se u memoriju za instrukcije upiše određeni program i nakon upisivanja pusti se da procesor izvrši taj program. Sledeći kodni listing implementira verifikacioni okruženje:

```
daZjYjq a]]]Å
mk] a]]]Çkl\ëdg_a[ëúúÿýÇYddÅ
mk] a]]]Çfme]ja[ëkl\ÇYddÅ
mk] kl\Çl]plagÇYddÅ
mk] ogjcÇlplëmladÇYddÅ

]flalq 4/0ë2)3#6ëlZ ak
ÈÈ hgjl ÝÞÅ
]f\ ]flalqÅ


Yj[`al][lmj] "]`YnagjYd g^ 4/0ë2)3#6ëlZ ak
   ÈÈ /h]jYf\ rY hjaklmh Yk]eZd]jkcge cg\m hjg_jYeY
   ^ad] 2)3#6ëafkljm[lagfk             Æ l]pl gh]f j]Y\ëeg\] ak
íÇÇ×ÇÇ×ÇÇ×ÇÇ×ÇÇ×2)3#6ëlZ×Ykk]eZdqë[g\]ÇlplíÅ
   ÈÈ 'dgZYdfa ka_fYda
   ka_fYd [dc                          Æ kl\ëdg_a[ Æð ìùìÅ
   ka_fYd j]k]l                        Æ kl\ëdg_a[Å
   ÈÈ 3a_fYda e]egjab] rY afkljmc[ab]
   ka_fYd ]fYëafkljëkÄ ]fZëafkljëk     Æ kl\ëdg_a[Å
   ka_fYd o]YëafkljëkÄ o]ZëafkljëkÆ kl\ëdg_a[ën][lgjÝü \goflg ùÞÅ
   ka_fYd Y\\jYëafkljëkÄ Y\\jZëafkljëk Æ kl\ëdg_a[ën][lgjÝĂ \goflg ùÞÅ
   ka_fYd \afYëafkljëkÄ \afZëafkljëk   Æ kl\ëdg_a[ën][lgjÝüú \goflg ùÞÅ
   ka_fYd \gmlYëafkljëkÄ \gmlZëafkljëk Æ kl\ëdg_a[ën][lgjÝüú \goflg ùÞÅ
   ka_fYd Y\\jZëafkljëüûëk             Æ kl\ëdg_a[ën][lgjÝüú \goflg ùÞÅ
   ÈÈ 3a_fYda e]egjab] rY hg\Ylc]
   ka_fYd ]fYë\YlYëkÄ ]fZë\YlYëk       Æ kl\ëdg_a[Å
   ka_fYd o]Yë\YlYëkÄ o]Zë\YlYëk       Æ kl\ëdg_a[ën][lgjÝü \goflg ùÞÅ
   ka_fYd Y\\jYë\YlYëkÄ Y\\jZë\YlYëk   Æ kl\ëdg_a[ën][lgjÝĂ \goflg ùÞÅ
   ka_fYd \afYë\YlYëkÄ \afZë\YlYëk     Æ kl\ëdg_a[ën][lgjÝüú \goflg ùÞÅ
   ka_fYd \gmlYë\YlYëkÄ \gmlZë\YlYëk   Æ kl\ëdg_a[ën][lgjÝüú \goflg ùÞÅ
   ka_fYd Y\\jYë\YlYëüûëk              Æ kl\ëdg_a[ën][lgjÝüú \goflg ùÞÅ

Z]_af

   ÈÈ -]egjabY rY afkljmc[ab]
   ÈÈ 0jaklmh ! Æ +gjakla k] rY afa[abYdarY[abm e]egjab] rY afkljmc[ab]
   ÈÈ 0jaklmh " Æ +gjakla k] rY [alYfb] afkljmc[abY g\ kljYf] hjg[]kgjY
   ÈÈ +gfklYfl]Æ
   ]fYëafkljëk   Ēđ ìúìÅ
   ]fZëafkljëk   Ēđ ìúìÅ
   Y\\jZëafkljëk Ēđ Y\\jZëafkljëüûëkÝĂ \goflg ùÞÅ
   o]Zëafkljëk   Ēđ Ýgl`]jk đē ìùìÞÅ
   \afZëafkljëk  Ēđ Ýgl`]jk đē ìùìÞÅ
   ÈÈ )fklYf[YÆ
   afkljm[lagfëe]e Æ ]flalq ogjcÇ"2!-ÝZ]`YnagjYdÞ
```

```
   _]f]ja[ eYhÝ7!$$2 đē úùÞ
   hgjl eYh Ý[dc      đē [dcÄ
            ÈÈ hjaklmh !
            ]fëYëa   đē ]fYëafkljëkÄ
            o]ëYëa   đē o]YëafkljëkÄ
            Y\\jëYëa đē Y\\jYëafkljëkÄ
            \YlYëYëa đē \afYëafkljëkÄ
            \YlYëYëg đē \gmlYëafkljëkÄ
            ÈÈ hjaklmh "
            ]fëZëa   đē ]fZëafkljëkÄ
            o]ëZëa   đē o]ZëafkljëkÄ
            Y\\jëZëa đē Y\\jZëafkljëkÄ
            \YlYëZëa đē \afZëafkljëkÄ
            \YlYëZëg đē \gmlZëafkljëkÞÅ


ÈÈ -]egjabY rY hg\Ylc]
ÈÈ 0jaklmh ! Æ +gjakla hjg[]kgj cYcg Za mhakanYg a [alYg hg\Ylc]
ÈÈ 0jaklmh " Æ .] cgjakla k]
ÈÈ +gfklYfl]Æ
Y\\jYë\YlYëk Ēđ Y\\jYë\YlYëüûëkÝĂ \goflg ùÞÅ
Y\\jZë\YlYëk Ēđ Ýgl`]jk đē ìùìÞÅ
\afZë\YlYëk  Ēđ Ýgl`]jk đē ìùìÞÅ
]fYë\YlYëk   Ēđ ìúìÅ
]fZë\YlYëk   Ēđ ìúìÅ
ÈÈ )fklYf[YÆ
\YlYëe]e Æ ]flalq ogjcÇ"2!-ÝZ]`YnagjYdÞ
   _]f]ja[ eYhÝ7!$$2 đē úùÞ
   hgjl eYh Ý[dc      đē [dcÄ
            ÈÈ hjaklmh !
            ]fëYëa   đē ]fYë\YlYëkÄ
            o]ëYëa   đē o]Yë\YlYëkÄ
            Y\\jëYëa đē Y\\jYë\YlYëkÄ
            \YlYëYëa đē \afYë\YlYëkÄ
            \YlYëYëg đē \gmlYë\YlYëkÄ
            ÈÈ hjaklmh "
            ]fëZëa   đē ]fZë\YlYëkÄ
            o]ëZëa   đē o]Zë\YlYëkÄ
            Y\\jëZëa đē Y\\jZë\YlYëkÄ
            \YlYëZëa đē \afZë\YlYëkÄ
            \YlYëZëg đē \gmlZë\YlYëkÞÅ


ÈÈ 4gh -g\md È 2)3#6 hjg[]kgj b]r_jg
4/0ë2)3#6ëú Æ ]flalq ogjcÇ4/0ë2)3#6
   hgjl eYh Ý
      [dc   đē [dcÄ
```

```
        j]k]l đē j]k]lÄ

        afkljëe]eëj]Y\ëa    đē \gmlZëafkljëkÄ
        afkljëe]eëY\\j]kkëg đē Y\\jZëafkljëüûëkÄ

        \YlYëe]eëo]ëg      đē o]Yë\YlYëkÄ
        \YlYëe]eëY\\j]kkëg đē Y\\jYë\YlYëüûëkÄ
        \YlYëe]eëj]Y\ëa    đē \gmlYë\YlYëkÄ
        \YlYëe]eëojal]ëg   đē \afYë\YlYëkÞÅ

   ÈÈ )fa[abYdarY[abY e]egjab] rY afkljmc[ab]
   ÈÈ 0jg_jYe cgba [] hjg[]kgj arnjkYnYla k] m[alYnY m e]egjabm
   j]Y\ë^ad]ëhjg[ Æ hjg[]kk
      nYjaYZd] jgo Æ daf]Å
      nYjaYZd] a   Æ afl]_]j Æđ ùÅ
   Z]_af
      j]k]l       Ēđ ìùìÅ
      o]Yëafkljëk Ēđ Ýgl`]jk đē ìúìÞÅ
      o`ad] Ýfgl ]f\^ad]Ý2)3#6ëafkljm[lagfkÞÞdggh
         j]Y\daf]Ý2)3#6ëafkljm[lagfkÄ jgoÞÅ
         Y\\jYëafkljëk Ēđ kl\ëdg_a[ën][lgjÝlgëmfka_f]\ÝaÄ úùÞÞÅ
         \afYëafkljëk  Ēđ lgëkl\ëdg_a[ën][lgjÝkljaf_ÝjgoÞÞÅ
         a             Æđ a č ýÅ
         oYal mflad jakaf_ë]\_]Ý[dcÞÅ
      ]f\ dgghÅ
      o]Yëafkljëk Ēđ Ýgl`]jk đē ìùìÞÅ
      j]k]l       Ēđ ìúì Y^l]j ûù fkÅ
      oYalÅ
   ]f\ hjg[]kkÅ

   ÈÈ cdgc ka_fYd _]f]jYlgj
   [dcëhjg[ Æ hjg[]kk
   Z]_af
      [dc Ēđ ìúìÄ ìùì Y^l]j úùù fkÅ
      oYal ^gj ûùù fkÅ
   ]f\ hjg[]kkÅ

]f\ Yj[`al][lmj]Å
```

Ako se pogleda kod, može se videti da su u njemu instancirani i povezani procesor i dve memorije. Pored toga, napisan je jednostavan kod koji vrši čitanje fajla u kome se nalazi program, njime se inicijalizuje memorija sa instrukcijama, odnosno vrši se upis instrukcija u memoriju.

Verifikacionom okruženju je neophodna VHDL implementacija RAM memorije i sledeći kodni listing predstavlja jednu vrstu implementacije:

```
daZjYjq a]]]Å
mk] a]]]Çkl\ëdg_a[ëúúÿýÇYddÅ
mk] a]]]Çfme]ja[ëkl\ÇYddÅ

]flalq "2!- ak
   _]f]ja[
      Ý
         7!$$2 Æ fYlmjYd Æđ úù
         ÞÅ
   hgjl
      Ý
         [dc      Æ af  kl\ëdg_a[Å
         ]fëYëa   Æ af  kl\ëdg_a[Å
         ]fëZëa   Æ af  kl\ëdg_a[Å
         \YlYëYëa Æ af  kl\ëdg_a[ën][lgjÝüú \goflg ùÞÅ
         \YlYëZëa Æ af  kl\ëdg_a[ën][lgjÝüú \goflg ùÞÅ
         Y\\jëYëa Æ af  kl\ëdg_a[ën][lgjÝ7!$$2 È ú \goflg ùÞÅ
         Y\\jëZëa Æ af  kl\ëdg_a[ën][lgjÝ7!$$2 È ú \goflg ùÞÅ
         o]ëYëa   Æ af  kl\ëdg_a[ën][lgjÝü \goflg ùÞÅ
         o]ëZëa   Æ af  kl\ëdg_a[ën][lgjÝü \goflg ùÞÅ
         \YlYëYëg Æ gml kl\ëdg_a[ën][lgjÝüú \goflg ùÞÅ
         \YlYëZëg Æ gml kl\ëdg_a[ën][lgjÝüú \goflg ùÞ
         ÞÅ

]f\ "2!-Å

Yj[`al][lmj] Z]`YnagjYd g^ "2!- ak lqh] jYeëlqh] ak YjjYqÝù lg ûãã7!$$2 È úÞ
                                    g^ kl\ëdg_a[ën][lgjÝĀ \goflg ùÞÅ ka_fYd jYeëk
Æ
                                                                        jYeëlqh]
Æđ Ýgl`]jk đē Ýgl`]jk đē ìùìÞÞÅ

Z]_af

   ÈÈ kaf`jgfa mhak
   hjg[]kkÝ[dcÞ
   Z]_af
      a^Ýjakaf_ë]\_]Ý[dcÞÞ l`]f
         a^Ý]fëYëa đ ìúìÞ l`]f
            a^Ýo]ëYëaÝüÞ đ ìúìÞ l`]f
               jYeëkÝlgëafl]_]jÝmfka_f]\ÝY\\jëYëaÞčüÞÞ Ēđ \YlYëYëaÝüú \goflg ûýÞÅ
            ]f\ a^Å
            a^Ýo]ëYëaÝûÞ đ ìúìÞ l`]f
               jYeëkÝlgëafl]_]jÝmfka_f]\ÝY\\jëYëaÞčûÞÞ Ēđ \YlYëYëaÝûü \goflg úÿÞÅ
            ]f\ a^Å
            a^Ýo]ëYëaÝúÞ đ ìúìÞ l`]f
               jYeëkÝlgëafl]_]jÝmfka_f]\ÝY\\jëYëaÞčúÞÞ Ēđ \YlYëYëaÝúþ \goflg āÞÅ
```

```
            ]f\ a^Å
            a^Ýo]ëYëaÝùÞ đ ìúìÞ l`]f
               jYeëkÝlgëafl]_]jÝmfka_f]\ÝY\\jëYëaÞÞÞ Ēđ \YlYëYëaÝĀ \goflg ùÞÅ
            ]f\ a^Å
         ]f\ a^Å
         a^Ý]fëZëa đ ìúìÞ l`]f
            a^Ýo]ëZëaÝüÞ đ ìúìÞ l`]f
               jYeëkÝlgëafl]_]jÝmfka_f]\ÝY\\jëZëaÞčüÞÞ Ēđ \YlYëZëaÝüú \goflg ûýÞÅ
            ]f\ a^Å
            a^Ýo]ëZëaÝûÞ đ ìúìÞ l`]f
               jYeëkÝlgëafl]_]jÝmfka_f]\ÝY\\jëZëaÞčûÞÞ Ēđ \YlYëZëaÝûü \goflg úÿÞÅ
            ]f\ a^Å
            a^Ýo]ëZëaÝúÞ đ ìúìÞ l`]f
               jYeëkÝlgëafl]_]jÝmfka_f]\ÝY\\jëZëaÞčúÞÞ Ēđ \YlYëZëaÝúþ \goflg āÞÅ
            ]f\ a^Å
            a^Ýo]ëZëaÝùÞ đ ìúìÞ l`]f
               jYeëkÝlgëafl]_]jÝmfka_f]\ÝY\\jëZëaÞÞÞ Ēđ \YlYëZëaÝĀ \goflg ùÞÅ
            ]f\ a^Å
         ]f\ a^Å
      ]f\ a^Å
   ]f\ hjg[]kkÅ

   ÈÈ Ykaf`jgfg [alYfb]
   hjg[]kkÝ]fëYëaÄ ]fëZëaÄ Y\\jëYëaÄ Y\\jëZëaÞ
   Z]_af
      a^Ý]fëYëa đ ìúìÞ l`]f
         \YlYëYëgÝüú \goflg ûýÞ Ēđ jYeëkÝlgëafl]_]jÝmfka_f]\ÝY\\jëYëaÞčüÞÞÅ
         \YlYëYëgÝûü \goflg úÿÞ Ēđ jYeëkÝlgëafl]_]jÝmfka_f]\ÝY\\jëYëaÞčûÞÞÅ
         \YlYëYëgÝúþ \goflg āÞ  Ēđ jYeëkÝlgëafl]_]jÝmfka_f]\ÝY\\jëYëaÞčúÞÞÅ
         \YlYëYëgÝĀ \goflg ùÞ   Ēđ jYeëkÝlgëafl]_]jÝmfka_f]\ÝY\\jëYëaÞÞÞÅ
      ]f\ a^Å
      a^Ý]fëZëa đ ìúìÞ l`]f
         \YlYëZëgÝüú \goflg ûýÞ Ēđ jYeëkÝlgëafl]_]jÝmfka_f]\ÝY\\jëZëaÞčüÞÞÅ
         \YlYëZëgÝûü \goflg úÿÞ Ēđ jYeëkÝlgëafl]_]jÝmfka_f]\ÝY\\jëZëaÞčûÞÞÅ
         \YlYëZëgÝúþ \goflg āÞ  Ēđ jYeëkÝlgëafl]_]jÝmfka_f]\ÝY\\jëZëaÞčúÞÞÅ
         \YlYëZëgÝĀ \goflg ùÞ   Ēđ jYeëkÝlgëafl]_]jÝmfka_f]\ÝY\\jëZëaÞÞÞÅ
      ]f\ a^Å
   ]f\ hjg[]kkÅ

]f\ Z]`YnagjYdÅ
```

3.9 Uputstvo za pokretanje simulacije

Uputstvo je namenjeno za korišćenje VIVADO 2019.1 alata i ukoliko se koristi neki drugi alat uputsvo verovatno neće moći da se primeni.

Prvi korak je pravljenje Vivado projekta. Da bi se taj proces olakšao, napisana je ***.tcl*** skripta koja kada se pokrene iz vivado alata automatski kreira projekat u koji su uključeni svi .vhd fajlovi. Skripta se pokreće tako što se u Vivado alatu izabere sledeće: ***Tools -> Run Tcl Script***. Nakon toga će se pojaviti prozor u kome je potrebno navesti putanju do RISCV.tcl skripte koja se nalazi u priloženim fajlovima.

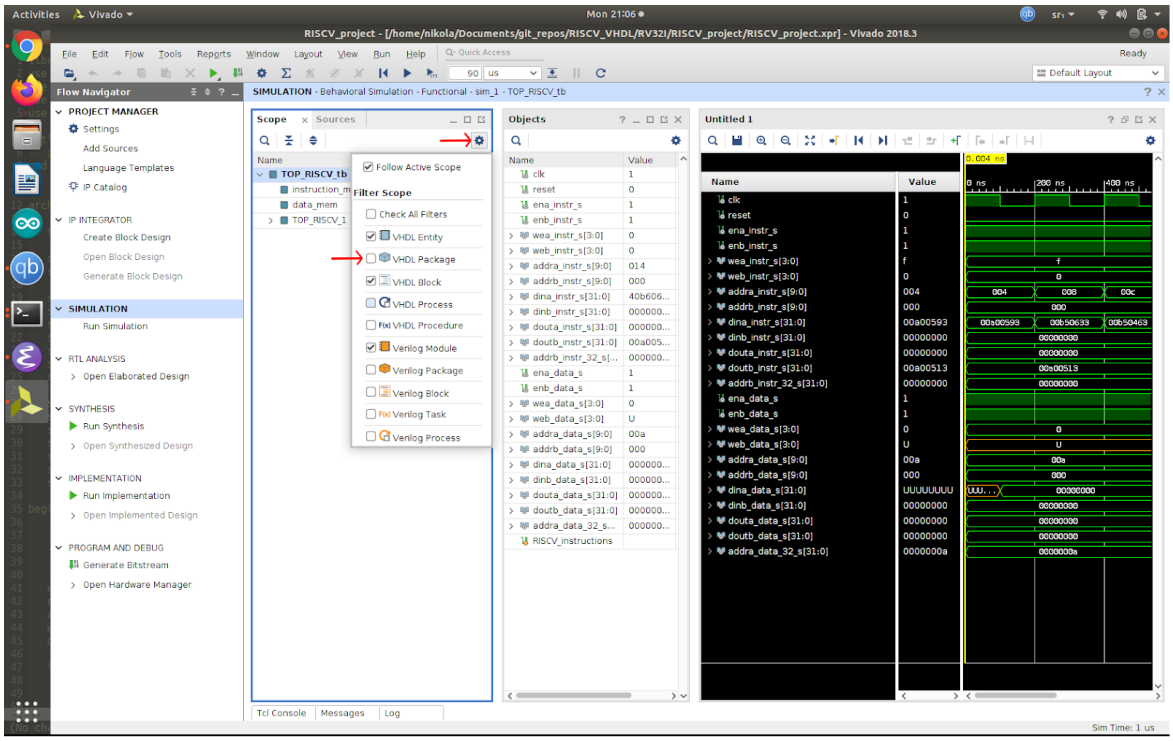
Kada skripta završi kreiranje projekta može se pokrenuti simulacija. U tački 3.8 je rečeno da verifikaciono okruženje prilikom pokretanja simulacije učitava u memoriju za instrukcije program koji je potrebno izvršiti. Fajl iz koga se program čita zove se ***assembly_code.txt*** i u njemu su upisane instrukcije u binarnom formatu. Svaka linija u fajlu predstavlja jednu instrukciju. Ukoliko čitalac želi da promeni program koji procesor izvršava, neophodno je da promeni ***assembly_code.txt*** fajl i upiše u njega druge instrukcije. Jednostavan način da se to uradi jeste da se iskoristi simulator opisan u vežbi 1, koji može da generiše mašinski kod u binarnom formatu.Kada se to uradi samo je potrebno kopirati binarni kod koji je generisao simulator u ***assembly_code.txt*** fajl i pokrenuti simulaciju.

Napomena: Ukoliko ne koristite verziju 2019.1 Vivado alata, može se desiti da se simulacija ne pokrene (iskočiće error). Ukoliko se to desi, neophodno je promeniti sledeću liniju koda u TOP_RISCV_tb fajlu i navesti apsolutnu putanju do assembly.txt fajla:

```
^ad] 2)3#6ëafkljm[lagfk             Æ l]pl gh]f j]Y\ëeg\] ak
íÇÇ×ÇÇ×ÇÇ×ÇÇ×ÇÇ×2)3#6ëlZ×Ykk]eZdqë[g\]ÇlplíÅ
                      Ø
                      | gornju liniju promeniti u donju (navesti svoju apsoultnu putanju)
                      |
^ad] 2)3#6ëafkljm[lagfk             Æ l]pl gh]f j]Y\ëeg\] ak
í×`ge]×facgdY×$g[me]flk×_alëj]hgk×2)3#6ë6($,×26üû)×2)3#6ëlZ×Ykk]eZdqë[g\]ÇlplíÅ
```

Da bi se zaključilo da procesor ispravno radi, potrebno je posmatrati stanje na I/O portovima procesora i njegove unutrašnje signale.

Takođe, ukoliko čitalac želi da posmatra promene stanja na signalima koji se nalaze unutar paketa neophodno je da uključi sledeće:



Slika 25. Vivado simulator. omogućavanje pristupa signalima unutar paketa.

4 Zadaci

1. Na osnovu prethodno opisanog, pokrenuti simulaciju u Vivado alatu i pokazati da procesor izvršava ispravno implementirani skup instrukcija.
2. Proširiti funkcionalnost aritmetičko logičke jedinice (ALU) tako da budu podržane sledeće instrukcije: SLL, SRL, XOR, SLT. Da bi procesor pravilno izvršio ove instrukcije neophodno je pored modifikacije ALU komponente modifikovati i ***alu_decoder*** komponentu tako da generiše ***alu_op_o*** shodno operaciji koju ALU treba da izvrši (npr. ***alu_op_o<= "00011"*** kada se izvršava ***xor*** instrukcija). Koju vrednost ***alu_op_o*** treba da dobije, u zavisnosti od operacije koju ALU treba da izvrši, možete naći u ***alu_ops_pkg.vhd fajlu***. Nakon proširenja, izvršiti simulaciju i pokazati da procesor izvršava instrukcije na ispravan način.
3. Proširiti skup instrukcija koje procesor podržava instrukcijom ***AUIPC (Add Upper Immediate to PC)***. Da bi se to omogućilo, potrebno je uraditi sledeće:

   a. Izmeniti ***immediate*** komponentu tako da vrši proširenje ***immediate*** polja instrukcije onako kako to nalaže format AUIPC instrukcije (opisano na prethodnim vežbama).
   b. Umetnuti 2-ulazni multiplekser ispred “a” ulaza ALU-a, gde se na jedan ulaz multipleksera dovodi rs1_data izlaz registarske banke, a na drugi ulaz vrednost PC registra. (Odredite sami šta se dovodi na selekcioni ulaz).
   c. Izmeniti kontrolnu jedinicu (prevashodno ***ctrl_decoder*** komponentu) tako da generiše kontrolne signale shodno prihvaćenoj AUIPC instrukciji. Ubaciti dodatne kontrolne signale, ukoliko je to neophodno.
4. Implementirati ostatak instrukcija uslovnog skoka (BNE, BLT, ...). Prilikom implementacije obratiti pažnju na ***funct3*** polje ***B*** formata instrukcija.
5. Implementirati ***jump and link*** (JAL) i ***jump and link register (JALR)*** instrukcije.
6. Implementirati ostatak instrukcija iz RV32I seta.