Fakultet Tehničkih Nauka
Novi Sad 2016

Projekat iz predmeta
Projektovanje složenih digitalnih sistema na temu:
DIZAJN JEDNOCIKLUSNOG MIPS PROCESORA

Mentor:
Rastislav Struharik

Student:
Tomislav Tumbas

Sadržaj

1 Zadatak

Implementirati i verifikovati jednociklusni procesor MIPS u jeziku za opis hardvera VHDL bez jedinice sa pokretnim zaredom i sledećim setom instrukcija:

1. add 2. addi 3. addu 4. addiu 5. sub 6. subu 7. and 8. andi 9. or 10. ori 11. nor	12. xor 13. xori 14. sllv 15. srlv 16. slt 17. sltu 18. slti 19. beq 20. j 21. lw 22. sw

2 Uvod

MIPS (skraćeno od Microprocessor without Interlocked Pipeline Stages) je procesor redukovanog instrukcijskog seta razvijen od strane MIPS computer systems-a 1985. godine. Nakon toga MIPS computer systems objavljuje još 5 generacija MIPS procesora. MIPS R3000 objavljen 1988. godine je prvi procesor koji doživljava veliku uspešnost. Napravljeno je preko milion komada i dostizali su frekvenciju od 40MHz. MIPS computer systems nastavlja da razvija svoje procesore i u drugoj polovini 90-tih godina dostižu zastupljenost u embedded svetu od 33%.

Danas MIPS procesore možemo pronaći u mrežnoj opremi, raznim konzolama (nitendo i plej stejšen) telefonima i drugim embeded uređajima.

3 MIPS

MIPS je Harvard arhitekture (razdvojena mu je memorija sa instrukcijama i podacima) i pored toga 32 registra generalne namene. Sve operacije su tipa register-register .

3.1 Instrukcije

MIPS poseduje 58 instrukcija i jos 19 pseudo instrukcija (instrukcije koje prepoznaje samo asembler i prevodi ih u instrukcije). Sve instrukcije podeljene su u tri grupe:

1. R instrukcije (operacije sa tri registra),
2. I instrukcije (operacije sa dva registra),
3. J instrukcije (operacije bez koristenja registarske banke) .

Type	-31-		Bits			-0-
R	opcode (6)	Rs (5)	Rt (5)	Rd (5)	shamt (5)	funct (6)
I	opcode (6)	Rs (5)	Rt (5)	immediate (16)		
J	opcode (6)	Address (26)				

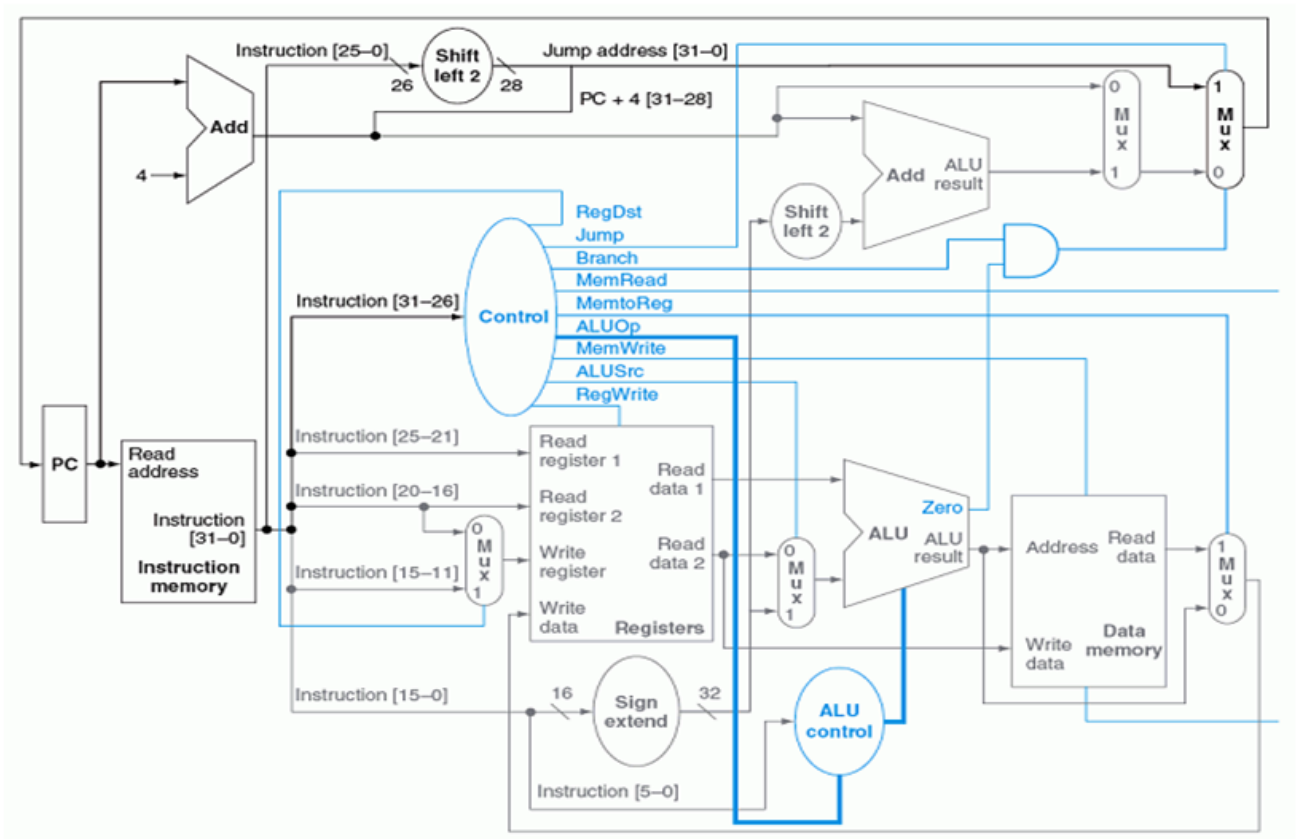
Svim instrukcijama je zajedničko da počinju sa opkodom. Opkod je deo instrukcije koji određuje koja će se operacija izvršiti. Iznimka su R instrukcije kod kojih opkod govori samo iz koje grupe operacija se bira operacija. Dok samu operaciju definiše polje funct iz instrukcije.

1. R instrukcije pored opkoda sadrže adrese tri registra (Rs, Rt i Rd) , shamt i funct polje. Oprand jedan nad kime će se izvršiti operacija predstavlja podatak iz registra banke sa adrese Rs. Oprand dva nad kime će se izvršiti operacija predstavlja podatak iz registarse banke sa adrese Rt. Rezultat operacije nad operandima jedan i dva smešta se u registarsku banku na adresu Rd. Polje shamt koristi se operacije pomeranja. Polje func kako je pre navedeno određuje operaciju koja se izvršava.
2. I instrukcije pored opkoda sadrži adrese dva registra (Rs i Rt) i polje immediate (16-bitnu konstantu). U ovom slučaju operand jedan je podatak iz registarske banke sa adrese Rs dok je drugi operand dat u samoj instrukciji tj. polje immediate predstavlja operand dva. Rezultat operacije smešta se u registarsku banku na adresu Rt.
3. J instrukcije pored opkoda sadrže samo adresu. Ovaj tip instrukcija obuhvata samo Jump instrukcije a polje address koristi se za određivanje sledeće vrednosti programskog brojača.

3.2 Arhitektura jednociklusnog MIPS-a

Jenociklusni MIPS sastoji se od:

1. programskog brojača i logike za odlčivanje naredne vrednosti,
2. kontrolne jedinice,
3. registarske banke,
4. aritmetičko logičke jedinice,
5. kotrolne jedinice aritmetičko logičke jedinice,
6. proširivača širine magistrale,
7. 4 multipeksera.



Arhitektura jednociklusnog MIPS-a

4 Implementacija

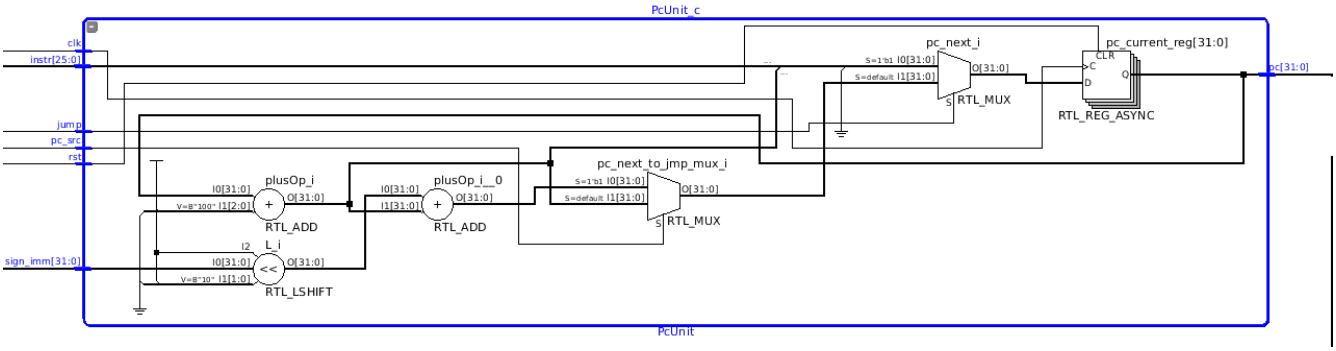
4.1 Programski brojač i logika za odlučivanje naredne vrednosti

Programski brojač je specijalan registar širine 32 bita. Čitanje vrednosti iz registra je asinhrona operacija, dok je upisivanje u registar je sinhrona operacija sa pozitivnom ivicom kloka.

Naredna vrednost programskog brojača bira se na osnovu signala Jump i PcSrc koji dolaze iz kontrolne jedinice. Ako je aktivan signal Jump znači da je trenutna instrukcija J te se naredna vrednost programskog brojača uzima od 26 do 0 bita instrukcije (pogledati tipove instrukcija i njihova polja).

Signal PcSrc dobija se logičkom operacijom I signala Branch iz kontrolne jedinice i signala Zero iz aritmetičko logičke jedinice. Ako je aktivan ovaj signal naredna vrednost programskog brojača dobija se kao rezultat sabiranja trenutnog programskog brojača, 4 i od 15-og do 0-tog bita instrkcije pomerenih za dva bita u levo.

Ako nisu aktivni Jump i PcSrc signali, naredna vrednost programskog brojača jednaka je zbiru trenutnog programskog brojača i 4. Ova vrednost predstavlja sledeću instrukciju u memoriji. Doadavanje broja 4 je iz razloga što je instrukcijska memorija bajt adresabilna, dok je jedna instrukcija 4 bajt.



Programski brojač i logika za odlučivanje naredne vrednosti

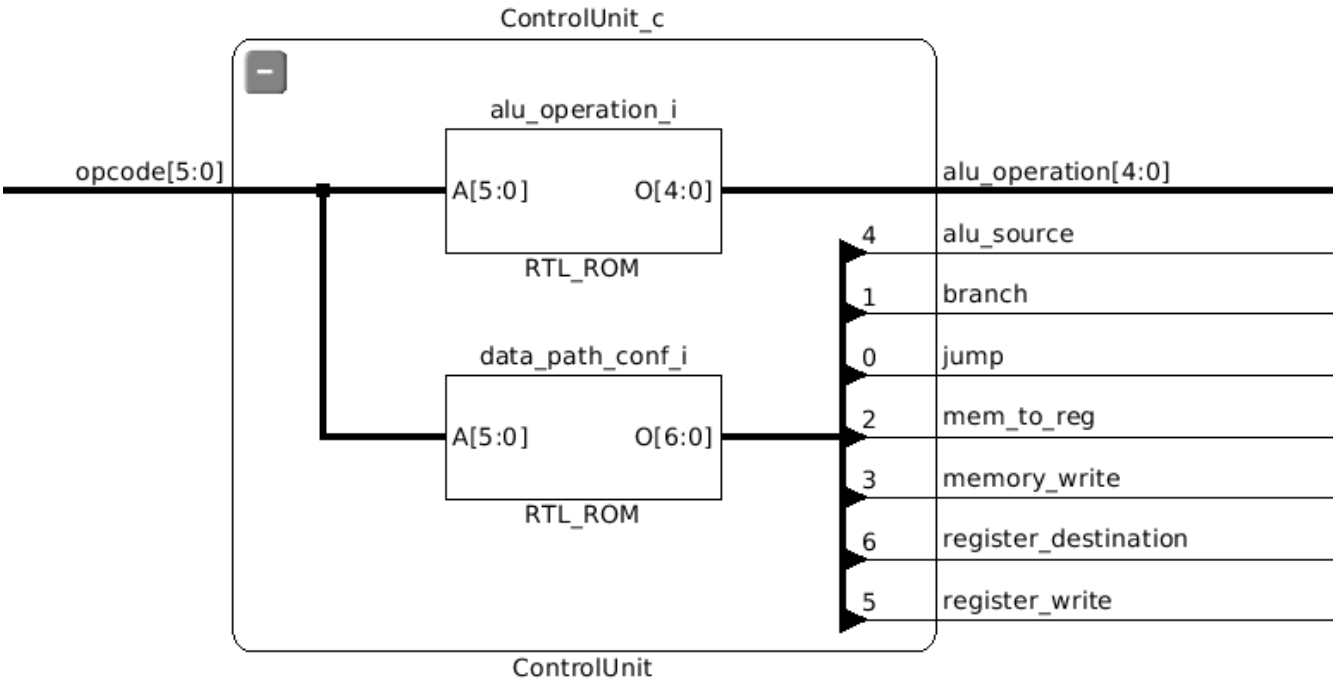
Programski brojač i logika za odlučivanje naredne vrednosti implementirani su kao kombinaciona mreža za određivanje narednog stanja i jedan D flip-flop implementiran kao sekvencijalna mreža. D flip-flop u slučaju pozitivne ivice kloka uzima izračunatu narednu vrednost i postavlja je na svoj izlaz što predstavlja trenutnu vrednost programskog brojača.

4.2 Kontrolna jedinica

Uloga kontrolne jedinice je da za svaku grupu instrukcija pravilno podesi putanju podataka (Data Path) i da dekoduje opkod instrukcije.

Kontrolna jedinica za određen opkod treba da postavi na izlaz određenu kombinaciju signala. Preporučena implemetacija ovakvih struktura je mikroprogramiranjem jer se time skraćuje vreme propagacije signala kroz jedinicu.

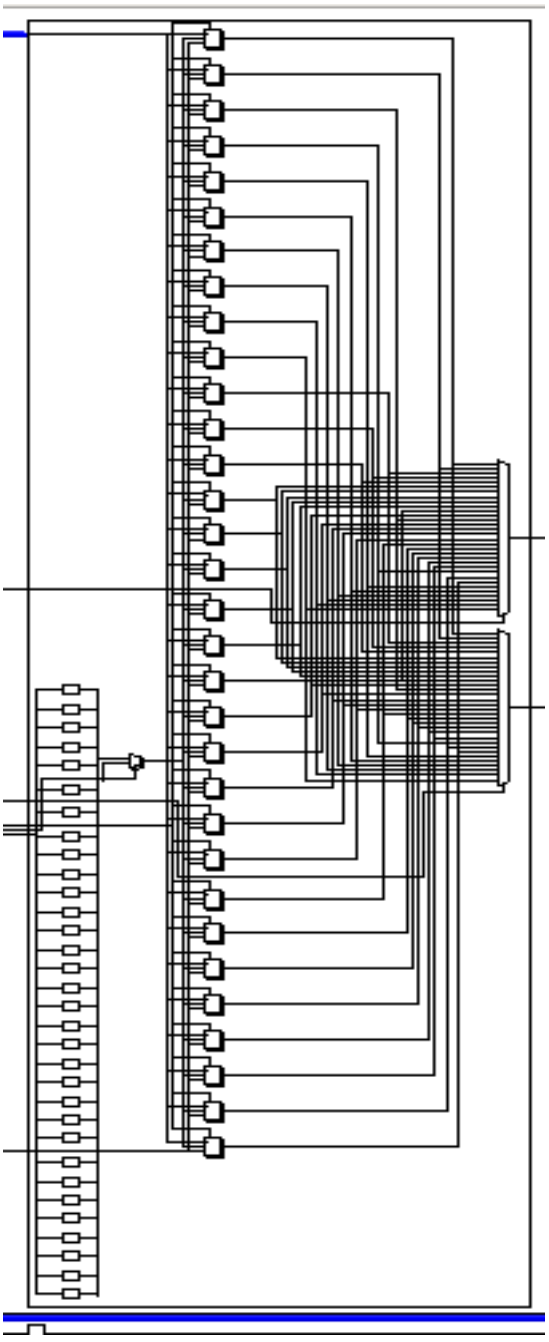
Kontrolna jedinica implemetirana je mikroprogramiranjem. Sastoji se od dva ROM-a, od kojih jedan dekoduje opkod i prosleđuje dalje kontrolnoj jedinici aritmetičko logičke jedinice. Dok drugi dekoduje opkod i postavlja signale kontrole putanje podataka (Data Path) na odgovarajuće vrednosti.



Kontrolna jedinica

4.3 Registarske banke

MIPS ima 32 registra opšte namene. Čitanje iz registra je asinhrona operacija dok je upisivanje u registar sinhrona. U jednom trenutku moguće je čitati vrednosti iz dva registra i upisivati u jedan.



Registarske banke

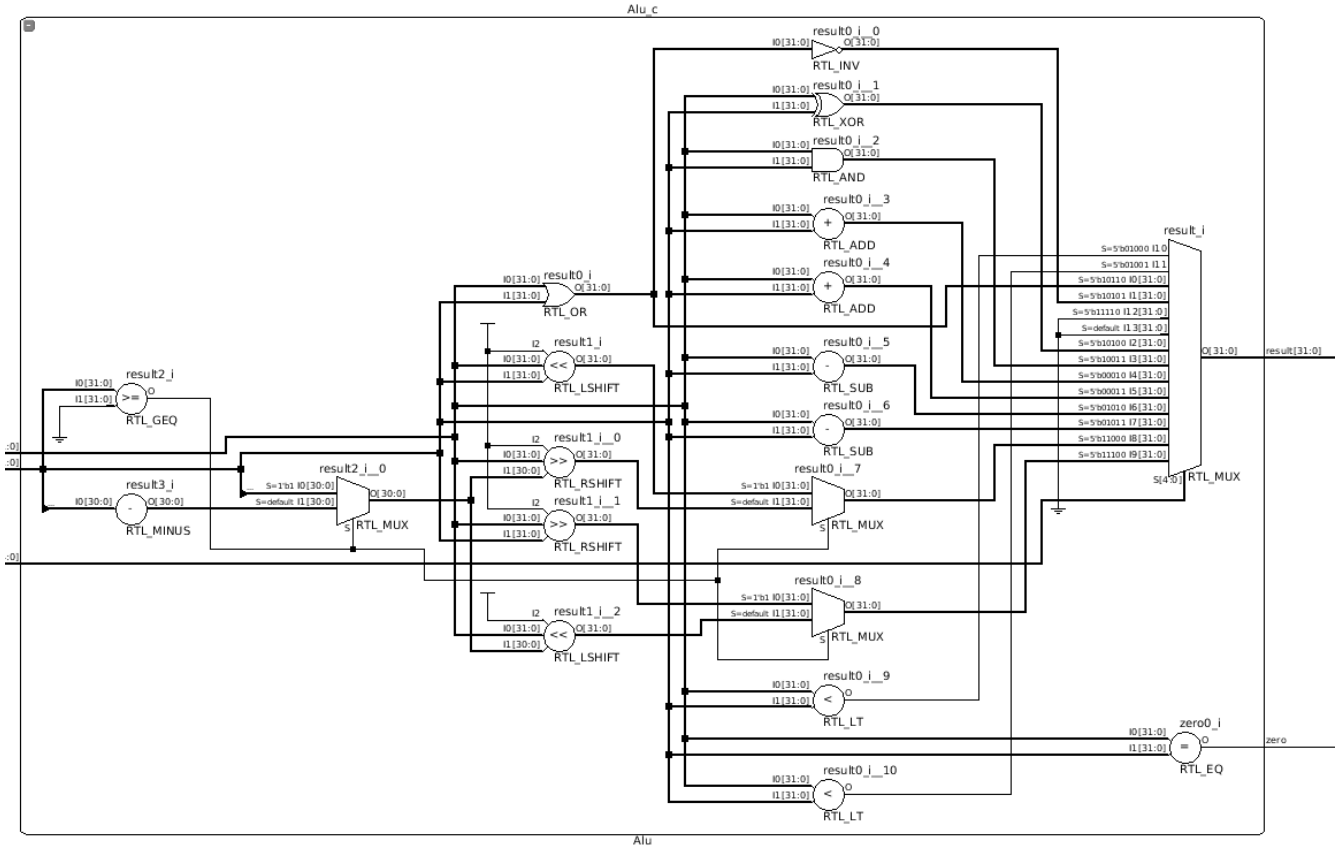
4.4 Aritmetičko logička jedinica (ALU)

Aritmetičko logička jedinica ima ulogu da nad ulaznim operandima jedan i dva izvršava zadatu operaciju. Pored toga ako su operandi jedan i dva jednaki, tada ALU setuje izlazni signal Zero. Signal Zero se kasnije korsiti u odlučiti koju vrednost programski brojač uzima u narednoj iteraciji. Operacije koje aritmetičko logička jedinica može da izvrši predstavlja podset intrukcijskog seta procesora. Primer: instrukcija add i addi imaju istu operaciju u ALU-u.

Operacija aritmetičko logičke jedinice mapirane na instrukcije koje podržava procesor date su u sledećoj tabeli:

		INSTRUKCIJA	PARAMETRI	OPRACIJA	ALU OPERACIJA
1	R	ADD	$Rd, $Rs, $Rt	$Rd = $Rs + $Rt	ADD
2	I	ADDI	$Rd, $Rs ,const16	$Rd = $Rs + const16	ADD
3	R	ADDU	$Rd, $Rs, $Rt	$Rd = $Rs + Rt (unsigned)	ADDU
4	I	ADDIU	$Rd, $Rs ,const16	$Rd = $Rs + const16 (unsigned)	ADDU
5	J	J	off16	Pc =Pc(31-18)::off26<<2	NOP
6	B	BEQ	$Rs $Rt off16	If($Rs == $Rt) Pc +4+= off16	NOP
7	B	BNE	$Rs $Rt off16	If($Rs != $Rt) Pc += off16	NOP
8	I	SW	$Rs off16($rt)	mem32($Rt + off16) = $Rs	ADD
9	I	LW	$Rs off16($rt)	$Rs = mem32($Rt + off16)	ADD
10	R	SUB	$Rd $Rs $Rt	$Rd = $Rs - $Rt	SUB
11	R	SUBU	$Rd $Rs $Rt	$Rd = $Rs - $Rt (unsigned)	SUBU
12	R	AND	$Rd $Rs $Rt	$Rd = $Rs & $Rt	AND
13	I	ANDI	$Rt $Rs const16	$Rt = $Rs & const16	AND
14	R	NOR	$Rd $Rs $Rt	$Rd = ~ ($Rs | $Rt)	NOR
15	R	OR	$Rd $Rs $Rt	$Rd = $Rs | $Rt	OR
16	I	ORI	$Rt $Rs const16	$Rd = $Rs | const16	OR
17	R	MULT	$Rs $Rt	LO = (($Rs * $Rt) << 32) >> 32; HI = ($Rs * $Rt) >> 32;	MULT
18	R	MULTU	$Rs $Rt	LO = (($Rs * $Rt) << 32) >> 32; HI = ($Rs * $Rt) >> 32;	MULTU
19	R	DIV	$Rs $Rt	LO = $Rs / $Rt HI = $Rs % $Rt	DIV
20	R	DIVU	$Rs $Rt	LO = $Rs / $Rt HI = $Rs % $Rt	DIVU
21	R	SLLV	$Rd $Rs $Rt	$Rd = $Rt << $Rs	SLL
22	R	SLRV	$Rd $Rs $Rt	$Rd = $Rt >> $Rs	SLR
23	R	XOR	$Rd $Rs $Rt	$Rd = $Rs ^ $tR	XOR

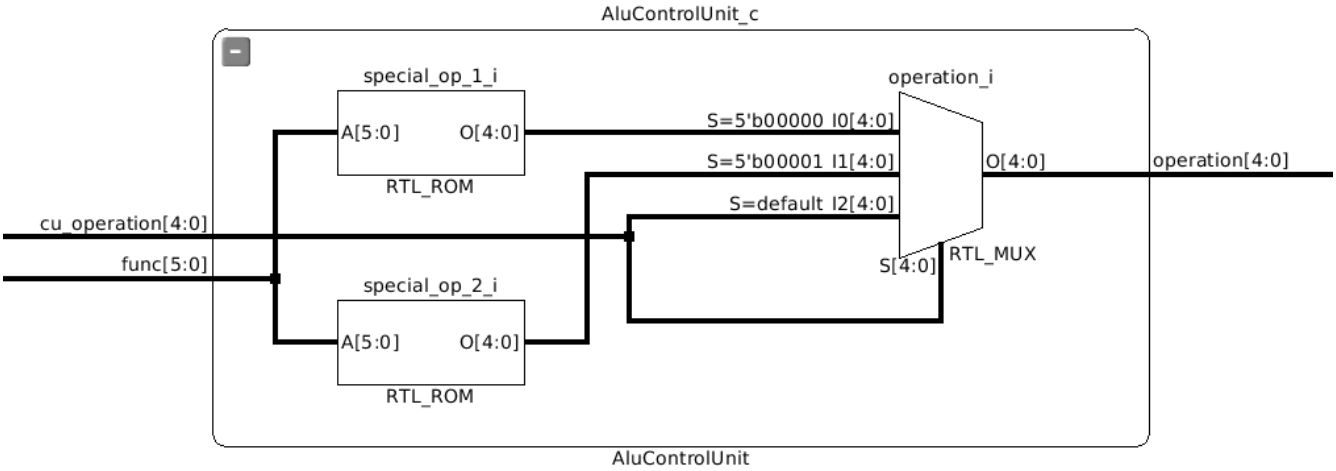
24	I	XORI	$Rt $Rs const16	$Rd = $Rs ^ Const16	XOR
25	R	SLT	$Rd $Rs $Rt	$Rd = ($Rs < $Rt)	SLT
26	R	SLTU	$Rd $Rs $Rt	$Rd = ($Rs < $Rt)	SLTU
27	I	SLTI	$Rt $Rs Const16	$Rt = ($Rs < Const16)	SLT



Aritmetičko logička jedinica

4.5 Kotrolna jedinica aritmetičko logičke jedinice

Kako sve instrukcije R tipa imaju dva zajednička opkoda, kontrolna jedinica u tom slučaju nije u mogućnosti da odredi koju operaciju ALU treba da izvrši. Uloga kontrolne jednice je da u tom slučaju pomoću polja func odredi koju tačno operaciju ALU treba da izvrši. Za oba R opkoda kontrolna jedinica ALU-a ima po jedan ROM koji određuje koja je to operacija. Konačna odluka koja će se operacija izvšiti donosi se na osnovu signala iz kontrolne jedinice.



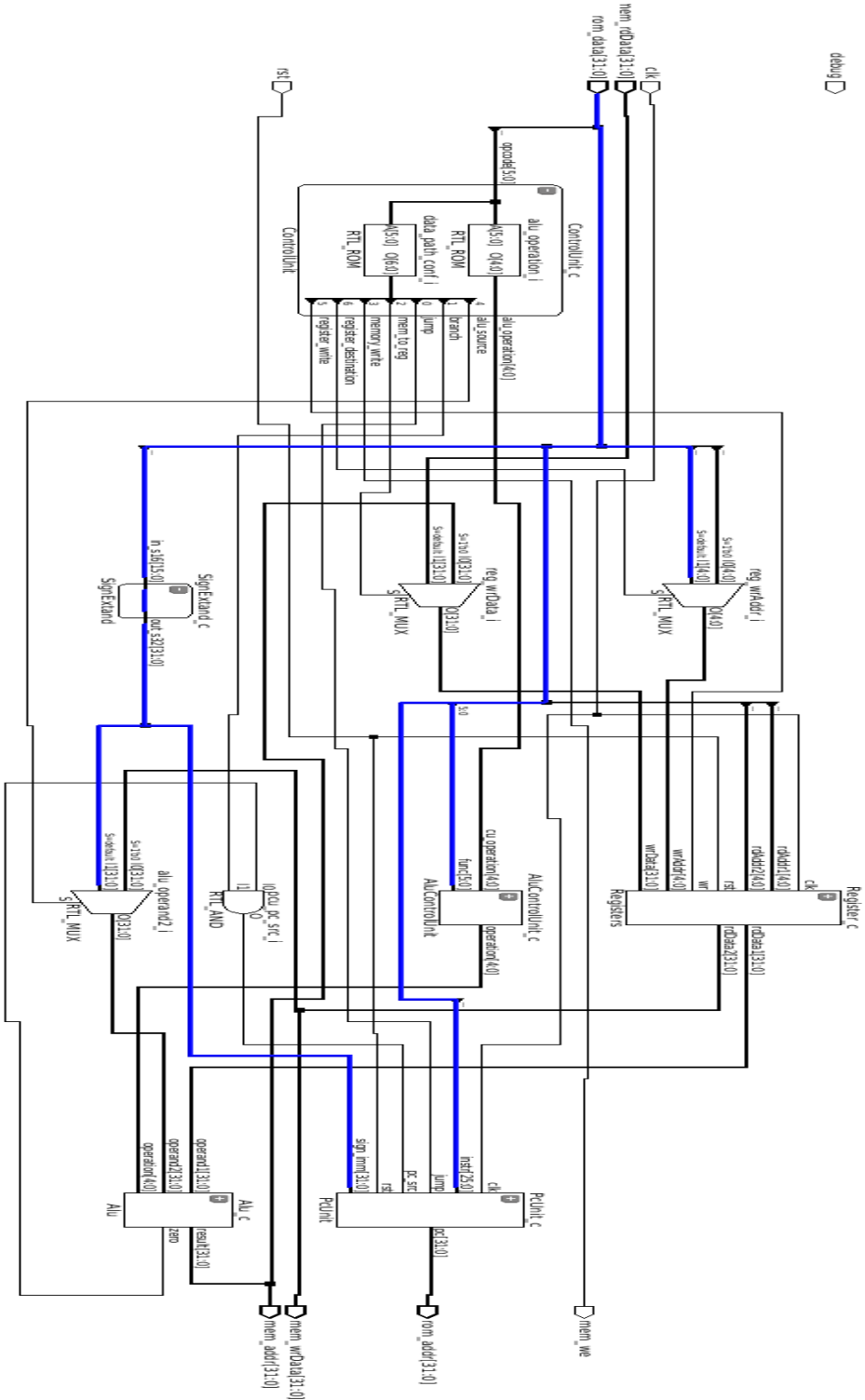
Kontrolna jedinca aritmetičko logičke jedinice

4.6 Proširivač magistrale

Uloga proširivača magistrale je da 16-bitnu vrednost proširi na 32-bitnu. Proširivač je sa predznakom, što znači da će u slučaju jedinice na poziciji 16-bitna na ulazu na izlazu na 16msb bita postaviti jedinice, a donjih 16 bita propustiti ulaz.

4.7 Procesor

Spajanjem svih komponenti i dodavanjem 4 multipleksera dobijamo procesor.



5 Verifikacija

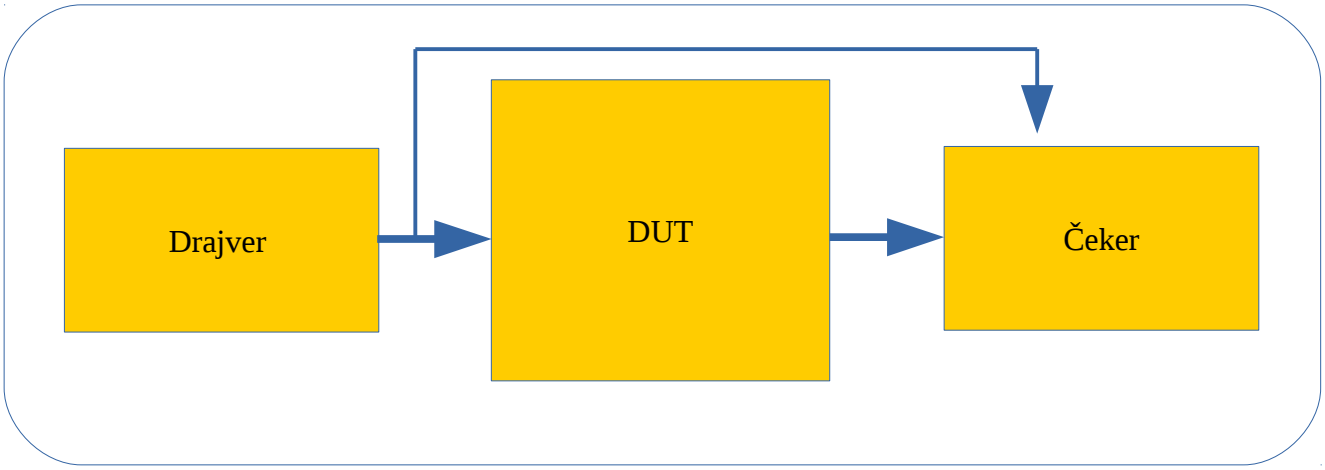
Verifikacija dizajna urađena je na dva nivoa:

- nivou pojedinačnih modula,
- na top nivou.

5.1 Verifikacija pojedinačnih modula

Za svaki modul u dizajnu razvijeno je posebno verifikaciono okruženje. Princip verifikacije je slučajna i direktna za specifične slučajeve, sa pristupom crne kutije. Jedno okruženje sastoji se od:

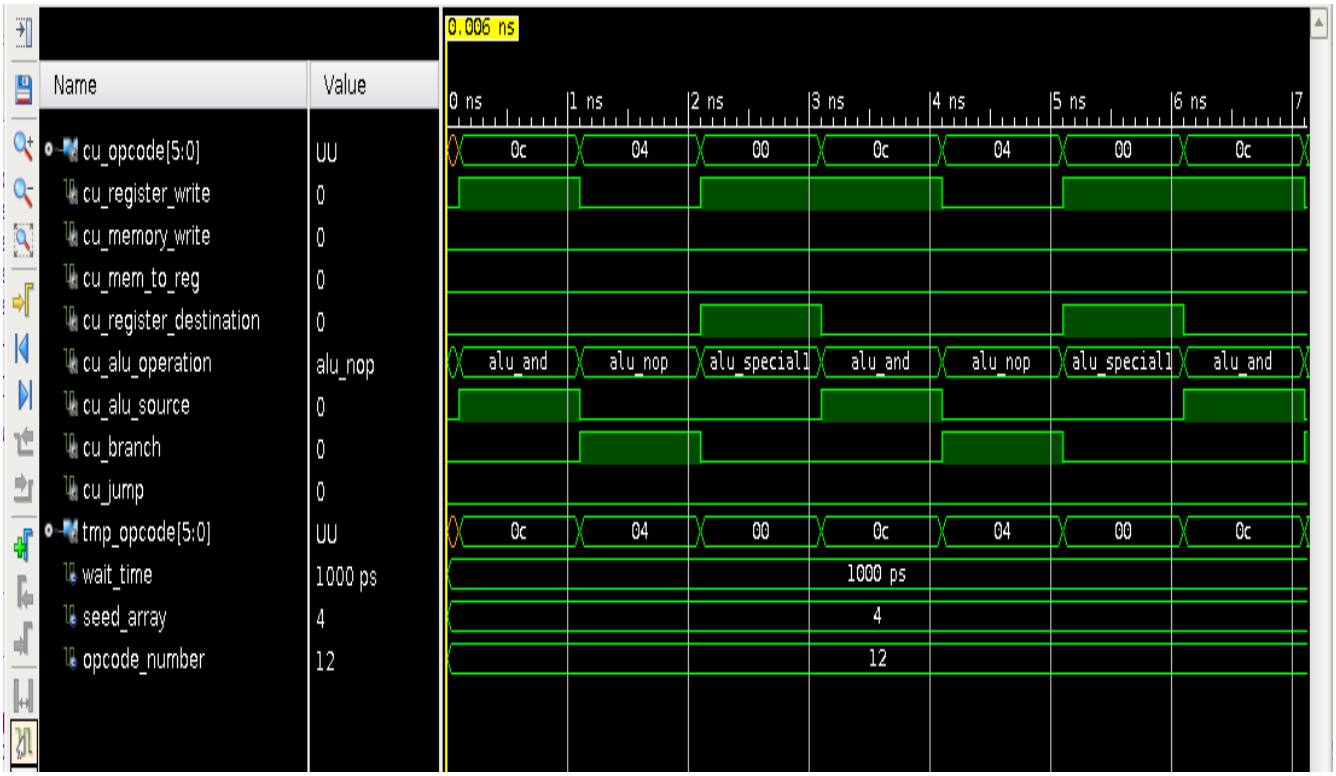
- drajvera
- čekera



Uloga drajvera je da generiše slučajne podatke i prosleđuje ih na ulaz modula koji se verifikuje (DUT-a). Kako je u nekim modulima potrebno da se pokriju neki specifični slučajevi postoje delovi drajvera koji generišu unapred određene podatke.

Uloga čekera je da proveravaju da li su izlazni podaci tačni, tj. da proveri da li modul radi ispravno. Za razliku od kompletnog verifikacionog okruženja u ovom slučaju ne postoje monitori i skorbord nego te uloge preuzima čeker.

5.2 Signali simulacija pojedinih modula



Simulacija kontrolne jedinice

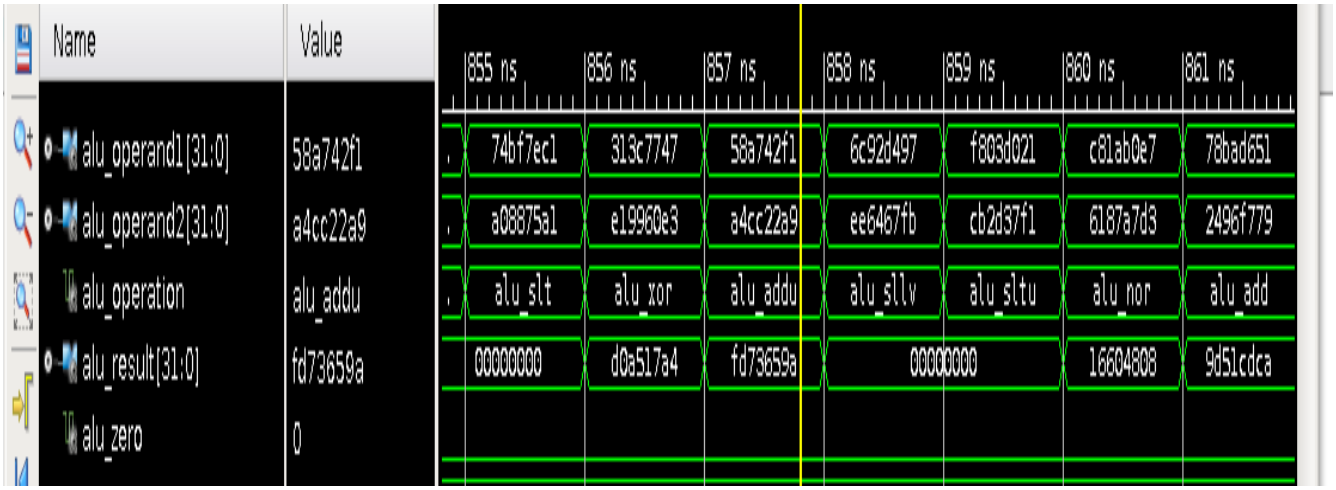
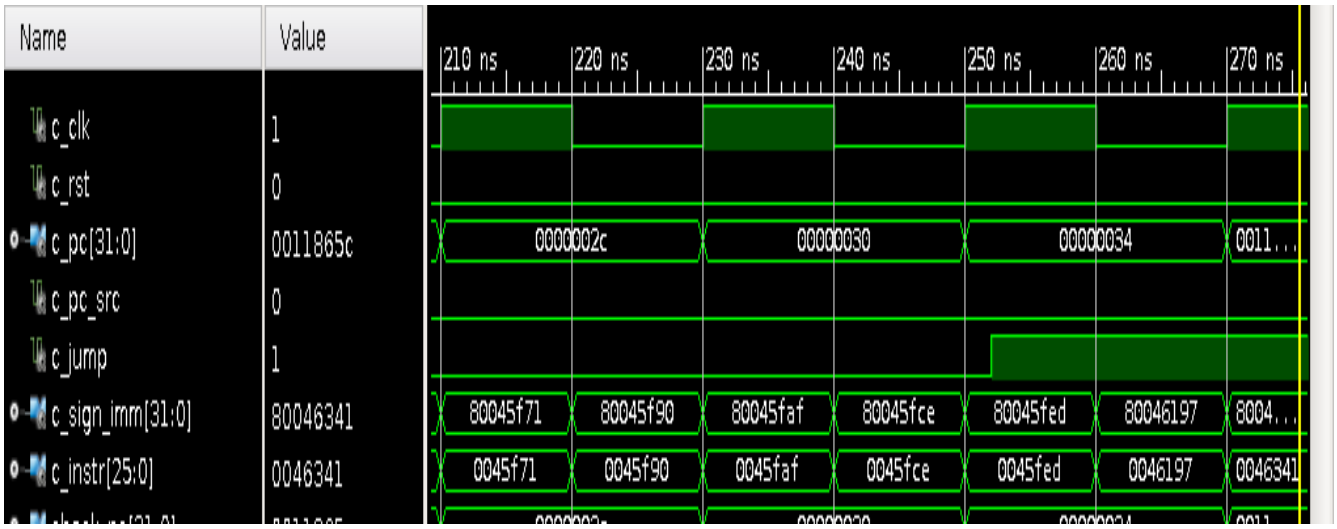


Illustration 1: Simulacija aritmetičko logičke jedinice



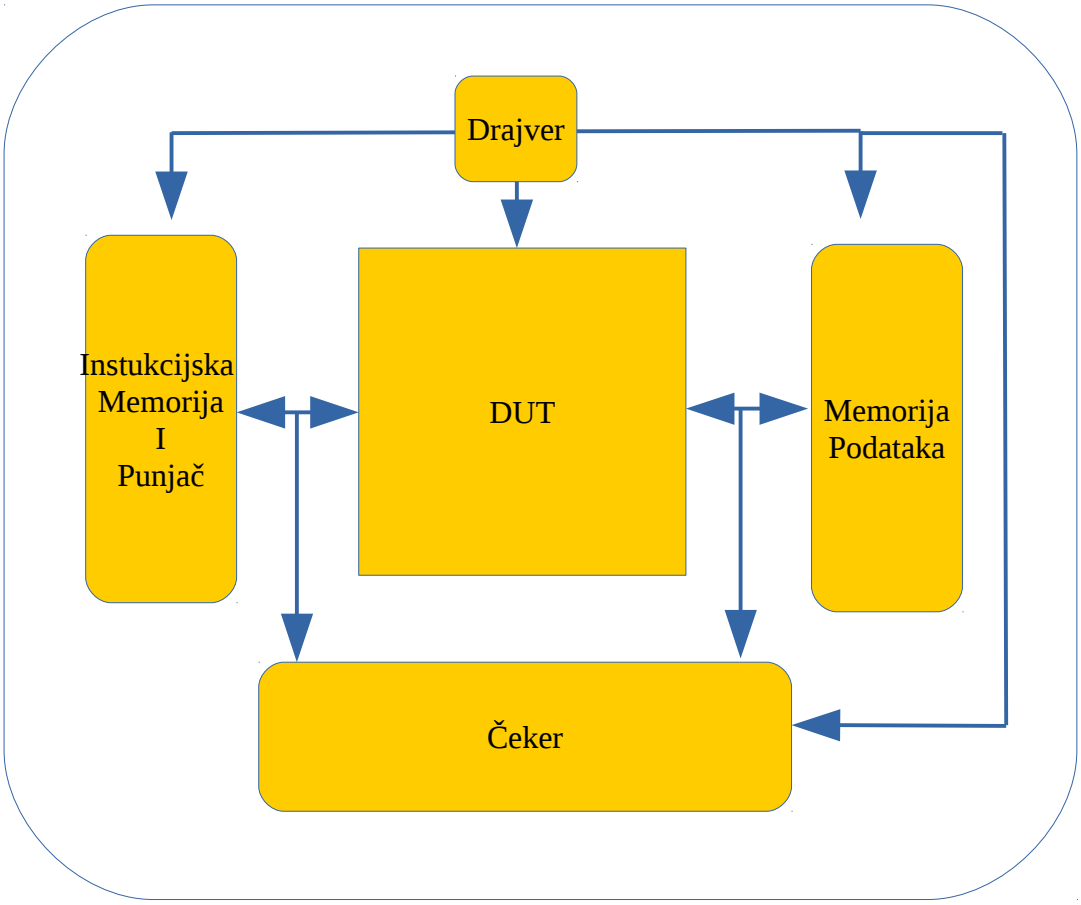
simulacija programskoh brojača i logike naredne vrednosti

5.3 Verifikacija top modula

Top modul predstavlja sam procesor sa potrebnim komponentama za simulaciju istog. Verifikacija top modula rađena je metodom zlatnih vektora i pristupom crne kutije. Ova metoda ostavlja moućnost provlačenja greške ali je njena verovatnoća mala zbog direktne i slučajne verifikacije svih pojedinačnih modula.

Verifikaciono okruženje procesora se sastoji od:

- instrukcijske memorije
- punjača instrukcijske memorije
- memorije podataka
- čekera
- drajvera

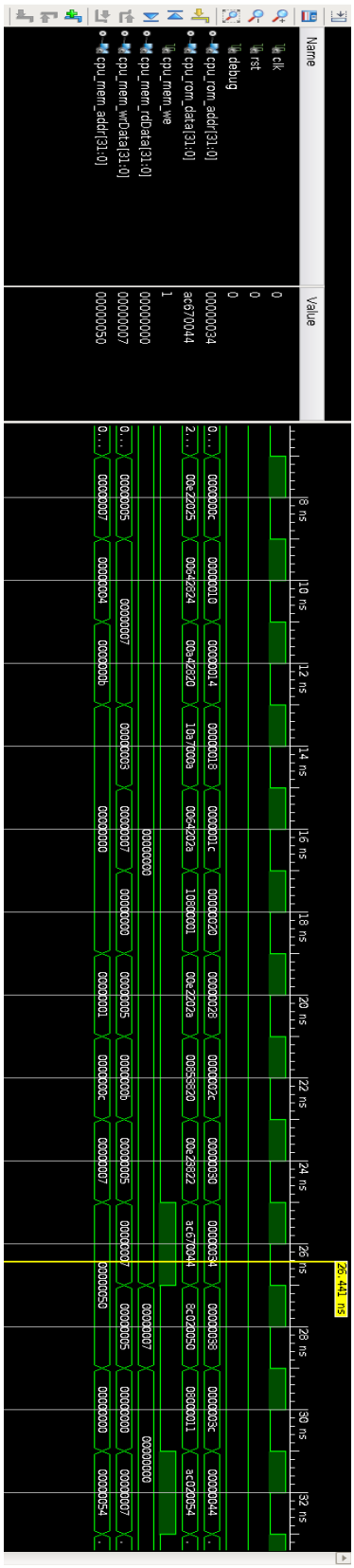


Uloga drajvera je da generiše klok i reset signal.

Uloga punjača instrukcijske memorije je da pri početku simlacije isčita zadati fajl sa programom i inicijalizuje memoriju skladno time.

Uloga čekera je da nadlgeda komunikaciju između procesora i memorija. Za pristupe meorijama proverava da li dešavaju upisi koji su u skladu sa programom.

5.4 Signali simulacije verifikacije top modula



6 Reference

[1] David Money Harris, Sarah L. Harris: *Digital Design and Computer Architecture.*

[2] John L. Hennessy, David A. Patterson: *Computer Architecture A Quantitative Approach*

[3] https://en.wikipedia.org/wiki/MIPS_instruction_set – 7.10.2016