

Sveučilište u Zagrebu
Fakultet elektrotehnike i računarstva

Digitalna logika

Laboratorijske vježbe korištenjem sklopovskih pomagala

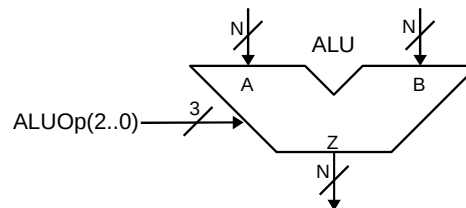
Upute za 4. laboratorijsku vježbu

Marko Zec

Prosinac 2013.

1 Zadatak: projektiranje aritmetičko-logičkog modula

Vaš je zadatak projektirati kombinacijski aritmetičko-logički modul (*Arithmetic-logic unit – ALU*) s dva N-bitna podatkovna ulaza (A i B) i jednim podatkovnim izlazom (Z), te upravljačkim ulazom za odabir aritmetičko-logičke operacije širine tri bita (ALUOp). Modul treba omogućiti obavljanje osam različitih operacija: cjelobrojno zbrajanje, oduzimanje, množenje, logički posmak u desno, logičko i, ili, ekskluzivno ili, te negirano ili. Kodiranje signala za odabir aritmetičko-logičke operacije ovisi o dvije znamenke najmanje težine JMBAG identifikacijskog broja, a zadano je tablicama na kraju ovih uputa.



Slika 1: sučelje aritmetičko-logičkog modula (ALU)

Modul ALU opišite jezikom VHDL, isključivo korištenjem konkurentnih naredbi i izraza (nije dozvoljeno korištenje blokova *process*). Širina ulaznih i izlaznih podatkovnih signala treba biti podesiva putem *generic* parametra prilikom instanciranja sklopa. Najmanja dozvoljena širina podatkovnih signala je dva bita.

Rad aritmetičko-logičkog modula ispitajte povezivanjem modula ALU u ispitni sklop koji se sastoji od bloka registara `reg_file` i upravljačkog modula `upravljac`.

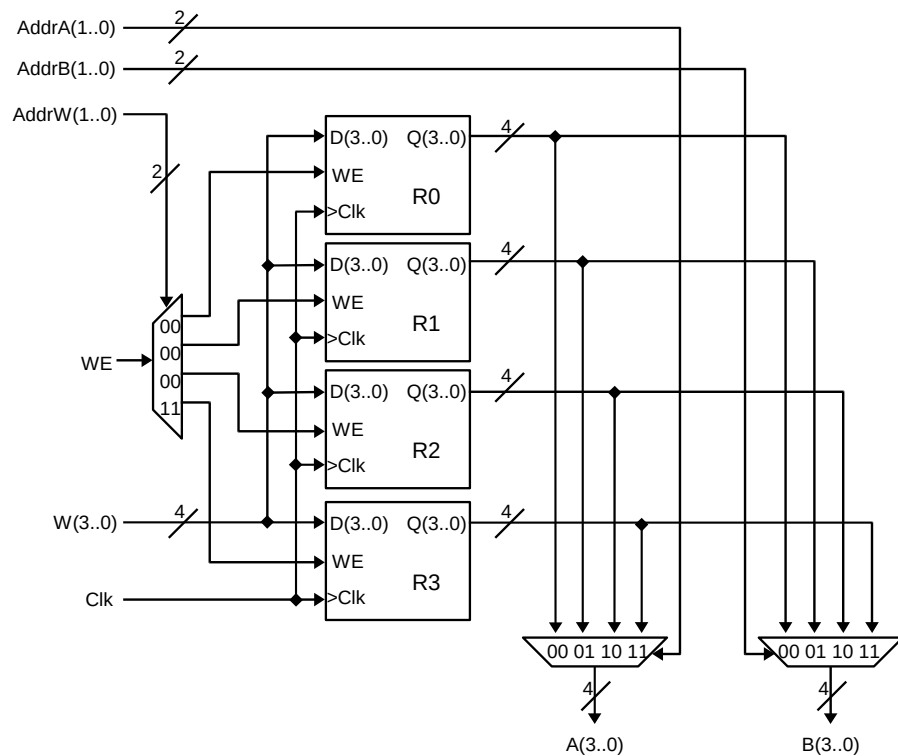
2 Ispitni sklop

Modul ALU potrebno je povezati s već gotovim sinkronim sekvencijskim (memorijskim) modulom `reg_file` u kojeg će se pohranjivati rezultati aritmetičko-logičkih operacija, te iz kojeg će se dohvaćati operandi A i B.

Modul `reg_file` izveden je kao dvodimenzionalno polje D bistabila okidanih rastućim bridom takta. D bistabili su temeljni memorijski elementi s jedobitnim podatkovnim ulazom (D), jednobitnim podatkovnim izlazom (Q), upravljačkim ulazom za omogućavanje pisanja (write enable – WE), te ulazom za signal takta (Clk). Izlaz iz bistabila Q biti će stabilan (nepromijenjen) neovisno o promjenama ulaznih signala D i WE, sve do trenutka u kojem signal takta Clk prelazi iz razine 0 na razinu 1 (tzv. rastući brid takta). Ukoliko u trenutku prelaska razine signala takta Clk iz 0 na 1 razina signala WE bude 1, bistabil će "zapamtiti" stanje na ulazu D i zadržati to stanje nepromijenjeno na izlazu Q do slijedeće pojave rastućeg brida takta. Instanca D bistabila se može opisati jezikom VHDL na slijedeći način:

```
--
-- D bistabil ili registar s write enable ulazom, okidan rastucim bridom signala clk.
-- Podatkovni ulaz D i izlaz Q moraju biti jednakog tipa i dimenzija.
--
Q <= D when rising_edge(clk) and WE = '1';
```

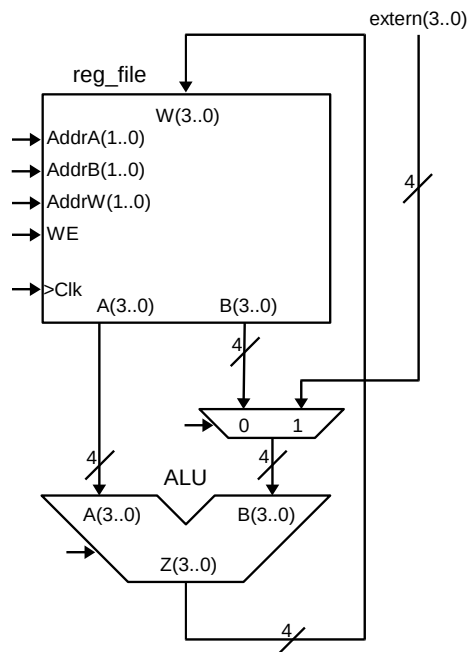
U modulu `reg_file` bistabili su grupirani u četiri četverobitna registra nazvana R0, R1, R2 i R3. Modul ima jedan višebitni ulazni podatkovni signal `W` čija će se vrijednost upisati u samo onaj od registara čija je adresa određena upravljačkim ulazom `AddrW` kod pojave rastućeg brida signala takta `Clk`, ukoliko u tom trenutku razina ulaza `WE` bude 1. Modul ima dva nezavisna višebitna podatkovna izlaza `A` i `B` na kojima se može očitati trenutno stanje registara čija je adresa određena asinkronim ulazima `AddrA` i `AddrB`. Struktura modula `reg_file` prikazana je shemom na slici 2:



Slika 2: struktura modula `reg_file`

Implementacija modula `reg_file` nalazi se u datoteci `"rf_4x4_1w_2r.vhd"` koju možete dohvatiti s web sjedišta laboratorijskih vježbi.

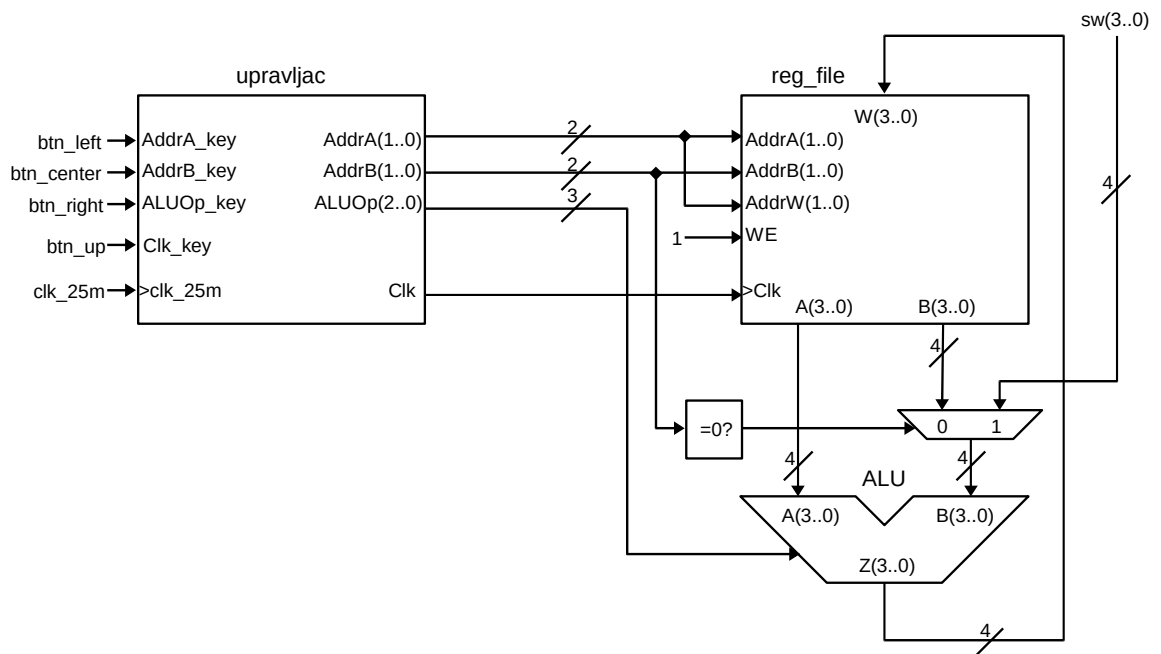
Blok-shemom na slici 3 prikazan je način na koji treba povezati module `ALU` i `reg_file`.



Slika 3: povezivanje modula ALU i reg_file

Podatkovni izlaz A iz modula reg_file potrebno je povezati direktno s podatkovnim ulazom A modula ALU, a podatkovni izlaz Z iz modula ALU potrebno je povezati direktno s podatkovnim ulazom W modula reg_file. Kako bi aritmetičko-logičke operacije osim nad registrima R0..R3, za koje je početno stanje nedefinirano, mogli izvoditi i s nekim drugim vrijednostima, na podatkovni ulaz B modula ALU povezuje se izlaz iz multipleksora pomoću kojeg se može odabrati drugi operand između izlaza B modula reg_file ili nekog drugog signala (u blok shemi vanjski signal je označen s *extern*).

U ispitni sklop uvodi se i već gotovi modul upravljač pomoću kojeg se može upravljati radom modula ALU (odabrati aritmetičko-logičku operaciju putem signala ALUOp) i radom modula reg_file (odabrati koji će od registara R0..R3 biti korišteni kao operandi aritmetičko-logičke operacije, te u koji će se registar pohraniti rezultat). Način povezivanja modula upravljač, reg_file i ALU prikazan je blok-shemom na slici 4.



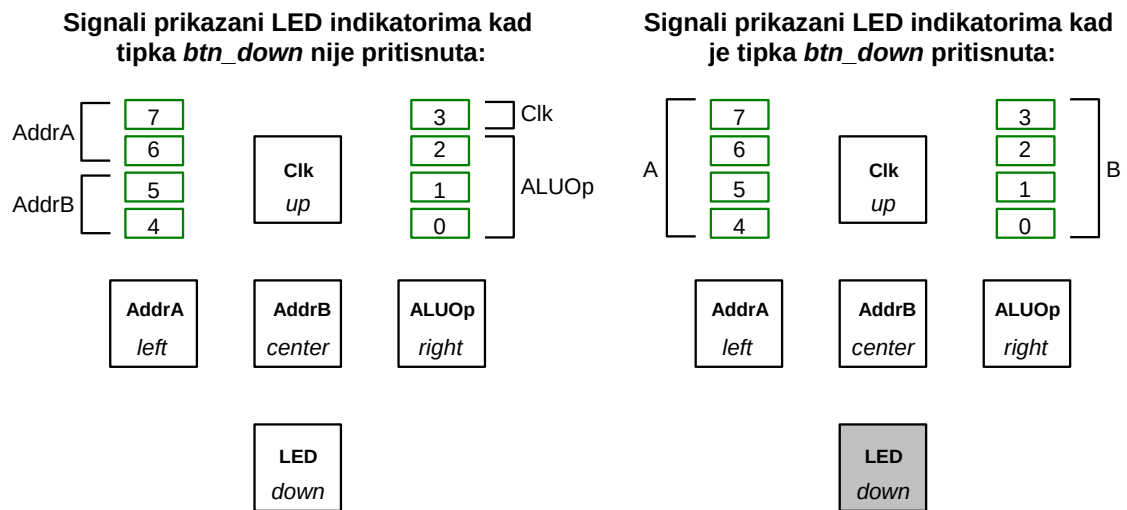
Slika 4: povezivanje modula ALU, reg_file i upravljac

Modul `upravljac` sinkroni je sklop koji omogućuje podešavanje stanja upravljačkih signala `AddrA`, `AddrB` i `ALUOp` pritiskom na tipke `btn_left`, `btn_center` i `btn_right`. Tipkom `btn_up` može se upravljati signalom takta (`Clk`). Rad modula `upravljac` sinkroniziran je signalom takta frekvencije 25 MHz (`clk_25m`) koji se dovodi iz oscilatora ugrađenog na razvojnu pločicu.

Iako modul `reg_file` omogućuje nezavisan odabir adrese registra u koji se upisuju podaci (`AddrW`) i adrese prvog podatkovnog izlaza (`AddrA`), u ispitnom sklopu na upravljačke ulaze `AddrA` i `AddrW` se dovodi isti signal, zbog čega će vrijednost registra koji je odabran kao prvi operand u svakom ciklusu takta `Clk` poprimiti novu vrijednost.

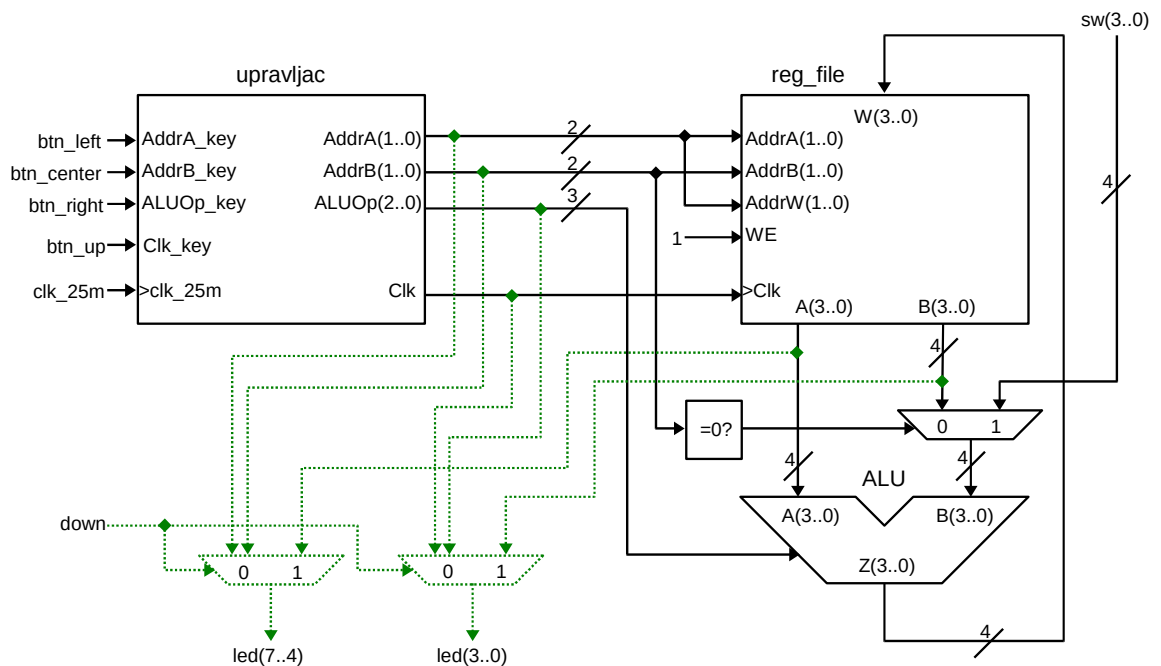
Na upravljački ulaz multipleksora za odabir operanda B modula ALU dovodi se izlaz iz komparatora koji na izlazu daje 1 ako je vrijednost signala `AddrB` jednaka nuli. Time je omogućeno da se za vrijednost `AddrB` = 0 kao operand B modula ALU odabere vanjski signal, čija vrijednost se postavlja 4-bitnim DIL mikroprekidačima na pločici (`sw`).

Kako bi se omogućio uvid u stanje podatkovnih i upravljačkih signala, najvažniji signali ispitnog sklopa povezuju se s LED indikatorima. Signali se na LED indikatore dovode preko multipleksora upravljanijh tipkom `btn_down` pomoću koje se može odabrati između prikaza podatkovnih signala (izlazi A i B iz modula `reg_file`) i upravljačkih signala (`AddrA`, `AddrB`, `ALUOp` i `Clk`). Multipleksori provode upravljačke signale na LED indikatore kad tipka `btn_down` nije pritisnuta, dok će za vrijeme dok je tipka `btn_down` pritisnuta na LED indikatorima biti prikazani podatkovni signali, kao što je prikazano shemom na slici 5.



Slika 5: prikaz signala na LED indikatorima zavisno od tipke `btn_down`

Konačna struktura ispitnog sklopa koja uključuje module ALU, `reg_file`, upravljac, te multipleksore za prikaz signala na LED indikatorima, prikazana je blok-shemom na slici 6.



Slika 6: multipleksori za odabir prikaza signala na LED indikatorima

3 Sinteza sklopa

Stvorite novi prazni direktorij na disku, te u njega pohranite slijedeće datoteke koje možete dohvatiti s web sjedišta laboratorijskih vježbi:

- **ulx2s.lpf** (definicije ulazno-izlaznih priključaka FPGA sklopa na pločici ULX2S)
- **datapath.vhd** (Predložak za izradu VHDL opisa ispitnog sklopa)

- **rf_4x4_1w_2r.vhd** (VHDL opis modula reg_file)
- **upravljac.vhd** (VHDL opis upravljačkog modula)

Ispitni sklop trebate konstruirati proširenjem predloška nazvanog datapath. U predlošku za izradu ispitnog sklopa datapath već su deklarirani svi potrebni ulazni i izlazni signali koji se povezuju s fizičkim priključcima i komponentama na razvojnoj pločici (tipke, prekidači, generator takta, LED indikatori), postavljene su instance modula upravljac i reg_file koji su povezani s nekim od ulaznih signala, te su instancirani multipleksori za odabir prikaza ispitnih signala na LED indikatorima.

Vaš je zadatak samostalno stvoriti implementaciju modula ALU, u postojeći sklop datapath postaviti instancu modula ALU i multipleksor za odabir ulaznog podatka B modula ALU, te sve module međusobno povezati prema shemi sa slike 6. Za izvedbu aritmetičkih operacija možete koristiti sljedeći VHDL predložak:

```
-- Zbrajanje - rezultat i operandi imaju iste dimenzije
Z <= A + B;

-- Oduzimanje - rezultat i operandi imaju iste dimenzije
Z <= A - B;

-- Množenje - rezultat mora biti dvostruko većih dimenzija od operanda
Z <= A * B;
```

Module reg_file i upravljac ne trebate i ne smijete modificirati!

4 Ispitivanje rada sklopa

Tablicom 2 zadan je slijed aritmetičkih i logičkih operacija koje trebate izvesti pomoću Vašeg sklopa i time ispitati njegov rad. Tablica je već djelomično popunjena, a Vaš je zadatak popuniti prazna polja odgovarajućim kodnim riječima za signale AddrA, AddrB, ALUOp i sw, te očekivane vrijednosti na izlazima A i B modula reg_file prije izvođenja operacije, te očekivanu vrijednost izlaza A nakon izvođenja operacije.

Na početku rada provjerite da li je sadržaj svih registara jednak nuli. Sadržaj registra može se postaviti na nulu operacijom $Rx = Rx \text{ and } 0000$. Za takvu operaciju upravljački signal ALUOp treba biti postavljen na kodnu riječ koja odabire operaciju "AND", a signal AddrB treba biti postavljen na 0, kako bi operand B modula ALU bio prosjopen na DIP prekidače (sw), koje sve treba postaviti na 0.

U prva tri koraka potrebno je u registre R1, R2 i R3 učitati vrijednosti koje odgovaraju znamenkama težine 5, 4 i 3 Vašeg JMBAG identifikacijskog broja. Ukoliko smo sigurni da je sadržaj svih registara prethodno postavljen na 0, učitavanje vrijednosti može se izvršiti operacijama $Rx = Rx + \textit{konstanta}$ ili $Rx = Rx \text{ or } \textit{konstanta}$, koju se kodira DIP prekidačima (sw).

Za izvršenje svakog koraka predviđenog tablicom 2 prvo treba podesiti sve upravljačke signale (AddrA, AddrB i ALUOp i sw), te provjeriti trenutno stanje na izlazima A i B iz modula reg_file pritiskom na tipku *btn_down*. Kada smo sigurni da su svi signali ispravno podešeni, pritiskom na tipku *btn_up* generirati će se impuls takta, čime će se rezultat Z odabrane aritmetičke odnosno logičke operacije upisati u registar određen upravljačkim signalom AddrA. Trenutno stanje na izlazima A i B modula reg_file u tablici 2 označeno je s A(t) i B(t), a novo stanje zapisano u registar određen adresom

AddrA nakon prolaska rastućeg brida impulsa takta označeno je s $A(t+1)$.

Definicije ponašanja modula ALU iz tablice u prilogu uputa koja odgovara Vašem JMBAG identifikatoru prepishite u tablicu 1.

JMBAG =

<u>ALUOp</u>	<u>Operacija</u>
000	
001	
010	
011	
100	
101	
110	
111	

Tablica 1: kodne riječi za odabir aritmetičko-logičkih operacija

<u>Operacija</u>	<u>AddrA</u>	<u>AddrB</u>	<u>ALUOp</u>	<u>A(t)</u>	<u>B(t)</u>	<u>sw</u>	<u>A(t+1)</u>
R1 = JMBAG(5)	0	00		0000	-		
R2 = JMBAG(4)	0	00		0000	-		
R3 = JMBAG(3)	11	00		0000	-		
R1 = R1 and JMBAG(2)	01	00					
R2 = R2 or JMBAG(1)							
R3 = R3 xor JMBAG(0)							
R1 = R1 + R2	01	10				-	
R2 = R2 - R3						-	
R1 = SRL R1, 2		00			-	0010	
R2 = R2 NOR 0000					-	0000	
R3 = R3 * 3							

Tablica 2: slijed operacija kojima treba ispitati rad sklopa

Operacije treba izvoditi točno zadanim slijedom!

Za pristupanje izvođenju 4. laboratorijske vježbe student mora imati ispunjene tablice 1 i 2 na ovoj stanici!

Nakon što ste ispitali rad sklopa, u sustav Ferko trebate prenijeti (upload) slijedeće datoteke:

- **datapath.vhd**
- **alu.vhd**
- **lab4.jed**

5 Dodatak: kodne riječi za aritmetičko-logičke operacije

Skraćenice aritmetičko-logičkih operacija imaju sljedeće značenje:

ADD: zbrajanje	$A + B$
SUB: oduzimanje	$A - B$
MUL: množenje	$A * B$
SRL: logički pomak operanda A u desno za B mjesta (shift right logical)	
AND: logičko I	A and B
OR: logičko ILI	A or B
XOR: logičko ekskluzivno ILI	A xor B
NOR: negirano logičko ILI	not (A or B)

Tablica određuje kodiranje aritmetičko-logičkih operacija zavisno od dvije znamenke najmanje težine JMBAG identifikacijskog broja:

JMBAG(1..0) = 0: ALUOp: Operacija: 0: sub 1: nor 2: or 3: xor 4: add 5: srl 6: and 7: mul	JMBAG(1..0) = 1: ALUOp: Operacija: 0: xor 1: nor 2: or 3: srl 4: sub 5: mul 6: add 7: and	JMBAG(1..0) = 2: ALUOp: Operacija: 0: srl 1: and 2: sub 3: mul 4: add 5: xor 6: nor 7: or	JMBAG(1..0) = 3: ALUOp: Operacija: 0: mul 1: add 2: sub 3: nor 4: or 5: and 6: xor 7: srl
JMBAG(1..0) = 4: ALUOp: Operacija: 0: add 1: sub 2: or 3: xor 4: and 5: nor 6: srl 7: mul	JMBAG(1..0) = 5: ALUOp: Operacija: 0: mul 1: sub 2: xor 3: srl 4: and 5: add 6: or 7: nor	JMBAG(1..0) = 6: ALUOp: Operacija: 0: srl 1: nor 2: mul 3: and 4: add 5: xor 6: sub 7: or	JMBAG(1..0) = 7: ALUOp: Operacija: 0: add 1: and 2: srl 3: or 4: xor 5: sub 6: mul 7: nor
JMBAG(1..0) = 8: ALUOp: Operacija: 0: nor 1: add 2: sub 3: and 4: xor 5: mul 6: or 7: srl	JMBAG(1..0) = 9: ALUOp: Operacija: 0: mul 1: or 2: add 3: and 4: srl 5: nor 6: xor 7: sub	JMBAG(1..0) = 10: ALUOp: Operacija: 0: srl 1: sub 2: or 3: nor 4: and 5: add 6: xor 7: mul	JMBAG(1..0) = 11: ALUOp: Operacija: 0: add 1: nor 2: srl 3: sub 4: mul 5: and 6: or 7: xor
JMBAG(1..0) = 12: ALUOp: Operacija: 0: srl 1: add 2: and 3: or 4: nor 5: sub 6: xor 7: mul	JMBAG(1..0) = 13: ALUOp: Operacija: 0: srl 1: xor 2: sub 3: and 4: or 5: add 6: nor 7: mul	JMBAG(1..0) = 14: ALUOp: Operacija: 0: xor 1: or 2: sub 3: srl 4: mul 5: and 6: add 7: nor	JMBAG(1..0) = 15: ALUOp: Operacija: 0: sub 1: or 2: srl 3: xor 4: mul 5: nor 6: and 7: add

JMBAG(1..0) = 16: ALUOp: Operacija: 0: xor 1: mul 2: nor 3: and 4: sub 5: add 6: or 7: srl	JMBAG(1..0) = 17: ALUOp: Operacija: 0: xor 1: sub 2: mul 3: srl 4: or 5: nor 6: and 7: add	JMBAG(1..0) = 18: ALUOp: Operacija: 0: srl 1: or 2: nor 3: add 4: sub 5: xor 6: mul 7: and	JMBAG(1..0) = 19: ALUOp: Operacija: 0: or 1: add 2: sub 3: xor 4: nor 5: and 6: mul 7: srl
JMBAG(1..0) = 20: ALUOp: Operacija: 0: or 1: and 2: mul 3: xor 4: add 5: srl 6: sub 7: nor	JMBAG(1..0) = 21: ALUOp: Operacija: 0: sub 1: xor 2: and 3: srl 4: mul 5: nor 6: or 7: add	JMBAG(1..0) = 22: ALUOp: Operacija: 0: sub 1: add 2: and 3: or 4: xor 5: mul 6: srl 7: nor	JMBAG(1..0) = 23: ALUOp: Operacija: 0: and 1: nor 2: xor 3: mul 4: or 5: sub 6: srl 7: add
JMBAG(1..0) = 24: ALUOp: Operacija: 0: nor 1: and 2: sub 3: or 4: add 5: srl 6: mul 7: xor	JMBAG(1..0) = 25: ALUOp: Operacija: 0: xor 1: srl 2: nor 3: or 4: sub 5: mul 6: add 7: and	JMBAG(1..0) = 26: ALUOp: Operacija: 0: mul 1: and 2: or 3: srl 4: nor 5: xor 6: add 7: sub	JMBAG(1..0) = 27: ALUOp: Operacija: 0: add 1: srl 2: or 3: xor 4: nor 5: mul 6: sub 7: and
JMBAG(1..0) = 28: ALUOp: Operacija: 0: or 1: xor 2: srl 3: add 4: nor 5: mul 6: sub 7: and	JMBAG(1..0) = 29: ALUOp: Operacija: 0: sub 1: and 2: or 3: srl 4: nor 5: add 6: xor 7: mul	JMBAG(1..0) = 30: ALUOp: Operacija: 0: nor 1: and 2: sub 3: mul 4: srl 5: or 6: add 7: xor	JMBAG(1..0) = 31: ALUOp: Operacija: 0: add 1: nor 2: mul 3: sub 4: or 5: xor 6: and 7: srl
JMBAG(1..0) = 32: ALUOp: Operacija: 0: srl 1: add 2: mul 3: and 4: sub 5: xor 6: nor 7: or	JMBAG(1..0) = 33: ALUOp: Operacija: 0: sub 1: and 2: or 3: nor 4: srl 5: mul 6: xor 7: add	JMBAG(1..0) = 34: ALUOp: Operacija: 0: and 1: sub 2: xor 3: or 4: add 5: mul 6: nor 7: srl	JMBAG(1..0) = 35: ALUOp: Operacija: 0: xor 1: sub 2: add 3: mul 4: or 5: srl 6: and 7: nor
JMBAG(1..0) = 36: ALUOp: Operacija: 0: mul 1: or 2: xor 3: sub 4: nor 5: and 6: srl 7: add	JMBAG(1..0) = 37: ALUOp: Operacija: 0: sub 1: mul 2: add 3: nor 4: xor 5: srl 6: or 7: and	JMBAG(1..0) = 38: ALUOp: Operacija: 0: and 1: nor 2: xor 3: sub 4: srl 5: or 6: add 7: mul	JMBAG(1..0) = 39: ALUOp: Operacija: 0: and 1: srl 2: or 3: sub 4: xor 5: nor 6: mul 7: add
JMBAG(1..0) = 40: ALUOp: Operacija: 0: xor 1: srl 2: and 3: nor 4: add 5: or 6: sub 7: mul	JMBAG(1..0) = 41: ALUOp: Operacija: 0: mul 1: srl 2: or 3: add 4: xor 5: and 6: sub 7: nor	JMBAG(1..0) = 42: ALUOp: Operacija: 0: sub 1: or 2: srl 3: and 4: nor 5: mul 6: add 7: xor	JMBAG(1..0) = 43: ALUOp: Operacija: 0: or 1: mul 2: srl 3: xor 4: sub 5: nor 6: add 7: and

JMBAG(1..0) = 44: ALUOp: Operacija: 0: sub 1: mul 2: srl 3: add 4: and 5: nor 6: xor 7: or	JMBAG(1..0) = 45: ALUOp: Operacija: 0: srl 1: xor 2: or 3: and 4: mul 5: sub 6: add 7: nor	JMBAG(1..0) = 46: ALUOp: Operacija: 0: and 1: or 2: mul 3: sub 4: srl 5: add 6: nor 7: xor	JMBAG(1..0) = 47: ALUOp: Operacija: 0: or 1: nor 2: xor 3: add 4: srl 5: mul 6: and 7: sub
JMBAG(1..0) = 48: ALUOp: Operacija: 0: mul 1: xor 2: nor 3: srl 4: or 5: add 6: sub 7: and	JMBAG(1..0) = 49: ALUOp: Operacija: 0: and 1: sub 2: srl 3: or 4: nor 5: xor 6: add 7: mul	JMBAG(1..0) = 50: ALUOp: Operacija: 0: add 1: and 2: xor 3: nor 4: or 5: sub 6: mul 7: srl	JMBAG(1..0) = 51: ALUOp: Operacija: 0: nor 1: mul 2: and 3: xor 4: sub 5: srl 6: or 7: add
JMBAG(1..0) = 52: ALUOp: Operacija: 0: or 1: sub 2: srl 3: nor 4: xor 5: and 6: mul 7: add	JMBAG(1..0) = 53: ALUOp: Operacija: 0: and 1: mul 2: xor 3: sub 4: add 5: srl 6: or 7: nor	JMBAG(1..0) = 54: ALUOp: Operacija: 0: xor 1: sub 2: mul 3: or 4: srl 5: nor 6: and 7: add	JMBAG(1..0) = 55: ALUOp: Operacija: 0: mul 1: nor 2: and 3: srl 4: or 5: add 6: xor 7: sub
JMBAG(1..0) = 56: ALUOp: Operacija: 0: or 1: mul 2: add 3: xor 4: nor 5: and 6: sub 7: srl	JMBAG(1..0) = 57: ALUOp: Operacija: 0: mul 1: add 2: srl 3: xor 4: and 5: sub 6: nor 7: or	JMBAG(1..0) = 58: ALUOp: Operacija: 0: or 1: nor 2: mul 3: and 4: srl 5: sub 6: xor 7: add	JMBAG(1..0) = 59: ALUOp: Operacija: 0: add 1: nor 2: or 3: srl 4: and 5: xor 6: mul 7: sub
JMBAG(1..0) = 60: ALUOp: Operacija: 0: srl 1: xor 2: mul 3: and 4: add 5: sub 6: nor 7: or	JMBAG(1..0) = 61: ALUOp: Operacija: 0: srl 1: and 2: add 3: mul 4: xor 5: sub 6: nor 7: or	JMBAG(1..0) = 62: ALUOp: Operacija: 0: and 1: or 2: srl 3: mul 4: nor 5: sub 6: add 7: xor	JMBAG(1..0) = 63: ALUOp: Operacija: 0: add 1: sub 2: srl 3: nor 4: mul 5: xor 6: and 7: or
JMBAG(1..0) = 64: ALUOp: Operacija: 0: and 1: nor 2: xor 3: or 4: mul 5: add 6: sub 7: srl	JMBAG(1..0) = 65: ALUOp: Operacija: 0: and 1: srl 2: mul 3: or 4: xor 5: sub 6: add 7: nor	JMBAG(1..0) = 66: ALUOp: Operacija: 0: or 1: xor 2: srl 3: add 4: mul 5: nor 6: sub 7: and	JMBAG(1..0) = 67: ALUOp: Operacija: 0: xor 1: or 2: and 3: sub 4: srl 5: add 6: nor 7: mul
JMBAG(1..0) = 68: ALUOp: Operacija: 0: mul 1: xor 2: srl 3: add 4: or 5: sub 6: nor 7: and	JMBAG(1..0) = 69: ALUOp: Operacija: 0: add 1: nor 2: sub 3: xor 4: and 5: srl 6: or 7: mul	JMBAG(1..0) = 70: ALUOp: Operacija: 0: srl 1: mul 2: and 3: nor 4: add 5: or 6: sub 7: xor	JMBAG(1..0) = 71: ALUOp: Operacija: 0: add 1: nor 2: xor 3: sub 4: and 5: mul 6: srl 7: or

JMBAG(1..0) = 72: ALUOp: Operacija: 0: sub 1: nor 2: xor 3: and 4: or 5: add 6: mul 7: srl	JMBAG(1..0) = 73: ALUOp: Operacija: 0: add 1: srl 2: xor 3: mul 4: nor 5: or 6: and 7: sub	JMBAG(1..0) = 74: ALUOp: Operacija: 0: add 1: and 2: srl 3: mul 4: nor 5: or 6: xor 7: sub	JMBAG(1..0) = 75: ALUOp: Operacija: 0: mul 1: or 2: srl 3: xor 4: nor 5: and 6: sub 7: add
JMBAG(1..0) = 76: ALUOp: Operacija: 0: srl 1: sub 2: mul 3: add 4: nor 5: xor 6: and 7: or	JMBAG(1..0) = 77: ALUOp: Operacija: 0: xor 1: mul 2: and 3: nor 4: or 5: sub 6: add 7: srl	JMBAG(1..0) = 78: ALUOp: Operacija: 0: srl 1: sub 2: or 3: xor 4: add 5: mul 6: nor 7: and	JMBAG(1..0) = 79: ALUOp: Operacija: 0: xor 1: mul 2: and 3: nor 4: sub 5: srl 6: add 7: or
JMBAG(1..0) = 80: ALUOp: Operacija: 0: nor 1: mul 2: and 3: sub 4: srl 5: add 6: xor 7: or	JMBAG(1..0) = 81: ALUOp: Operacija: 0: add 1: nor 2: xor 3: srl 4: sub 5: or 6: and 7: mul	JMBAG(1..0) = 82: ALUOp: Operacija: 0: or 1: srl 2: xor 3: add 4: mul 5: sub 6: nor 7: and	JMBAG(1..0) = 83: ALUOp: Operacija: 0: mul 1: and 2: add 3: srl 4: xor 5: nor 6: sub 7: or
JMBAG(1..0) = 84: ALUOp: Operacija: 0: mul 1: sub 2: xor 3: add 4: or 5: and 6: nor 7: srl	JMBAG(1..0) = 85: ALUOp: Operacija: 0: sub 1: and 2: xor 3: add 4: nor 5: srl 6: mul 7: or	JMBAG(1..0) = 86: ALUOp: Operacija: 0: and 1: srl 2: nor 3: sub 4: add 5: mul 6: xor 7: or	JMBAG(1..0) = 87: ALUOp: Operacija: 0: mul 1: add 2: nor 3: xor 4: and 5: sub 6: srl 7: or
JMBAG(1..0) = 88: ALUOp: Operacija: 0: and 1: nor 2: srl 3: or 4: add 5: xor 6: sub 7: mul	JMBAG(1..0) = 89: ALUOp: Operacija: 0: mul 1: nor 2: sub 3: and 4: srl 5: add 6: or 7: xor	JMBAG(1..0) = 90: ALUOp: Operacija: 0: add 1: or 2: and 3: sub 4: srl 5: mul 6: xor 7: nor	JMBAG(1..0) = 91: ALUOp: Operacija: 0: xor 1: nor 2: add 3: sub 4: srl 5: and 6: mul 7: or
JMBAG(1..0) = 92: ALUOp: Operacija: 0: xor 1: and 2: nor 3: mul 4: sub 5: srl 6: add 7: or	JMBAG(1..0) = 93: ALUOp: Operacija: 0: srl 1: mul 2: or 3: sub 4: add 5: xor 6: and 7: nor	JMBAG(1..0) = 94: ALUOp: Operacija: 0: xor 1: sub 2: or 3: and 4: srl 5: mul 6: add 7: nor	JMBAG(1..0) = 95: ALUOp: Operacija: 0: nor 1: add 2: srl 3: and 4: sub 5: mul 6: or 7: xor
JMBAG(1..0) = 96: ALUOp: Operacija: 0: xor 1: mul 2: srl 3: or 4: add 5: and 6: sub 7: nor	JMBAG(1..0) = 97: ALUOp: Operacija: 0: and 1: nor 2: sub 3: xor 4: srl 5: add 6: or 7: mul	JMBAG(1..0) = 98: ALUOp: Operacija: 0: mul 1: nor 2: xor 3: or 4: add 5: sub 6: and 7: srl	JMBAG(1..0) = 99: ALUOp: Operacija: 0: nor 1: add 2: mul 3: srl 4: sub 5: xor 6: and 7: or