Projektovanje procesora – proširenje skupa instrukcija

# Pristup memoriji za podatke

Omogućiti pristup memoriji za podatke preko instrukcija LOAD i STORE. Opis instrukcija je dat u Tabeli 1-1.

Tabela 1-1. Instrukcije pristupa memoriji

|  |  |  |
| --- | --- | --- |
| **Operacija** | **Funkcija** | **Kod instrukcije** |
| ld Rz,Ry | RZ ← [[RY]] | 100000 |
| st Ry,Rx | [RY] ← [RX] | 110000 |

Instrukcija LOAD upisuje u registar RZ sadržaj koji se nalazi u memoriji podataka na adresi koja piše u registru RY (zbog toga je u opisu funkcije stavljeno dvostruko dereferenciranje registra RY). Izvršavanje instrukcije se odvija u sledeće 4 faze:

1. Instrukcija se učitava u IR.
2. Multiplekser MUXB selektuje registar RY i njegova vrednost se prosleđuje na izlaz oADDR; u istom taktu vrednost iz memorije se pojavljuje na ulazu iDATA, pošto se čitanje memorije vrši kombinaciono; multiplekser MUXA selektuje vrednost iDATA.
3. ALU propušta vrednost iz registra A koja je jednaka vrednosti iDATA.
4. Vrednost iDATA se, iz registra C, upisuje u registar RZ.

Instrukcija STORE upisuje u memoriju, na adresu koja piše u registru RY, vrednost koja piše u registru RX. Izvršavanje instrukcije se odvija u sledeće 4 faze:

1. Instrukcija se učitava u IR.
2. Multiplekser MUXB selektuje registar RY i njegova vrednost se prosleđuje na izlaz oADDR. Multiplekser MUXA selektuje registar RX.
3. ALU propušta vrednost iz registra A.
4. Vrednost iz registra C se upisuje u memoriju (postavlja se signal dozvole sMEM\_WE).

Zbog potrebe instrukcije STORE, a da ne bi morali menjati arhitekturu procesora, promeniti signal selekcije multipleksera MUXB tako da on zadržava vrednost definisanu trenutnom instrukcijom i u fazama EXECUTE i WRITE\_BACK. Takođe, ukoliko je potrebno, promeniti generisanje signala oREG\_WE u WRITE\_BACK fazi tako da se tokom instrukcije STORE vrednost ne upisuje ni u jedan registar.

# Test program

Data je funkcija:

U instrukcijskoj memoriji napisati program za računanje vrednosti i proveriti rad programa u simulatoru.

Rezultat sačuvati na memorijskoj lokaciji na adresi 0x0000, a nakon toga vrednost sa te lokacije učitati u registar opšte namene. Proveru rada izvršiti simulacijom.

# Sinteza procesora

Procesor ćemo konačno i sintetisati za TLL5000 platformu. Kako bi proverili rad procesora u realnom vremenu na platformi, na LCD ćemo ispisivati vrednosti programskog brojača, instrukcionog registra, trenutnu fazu izvršenja instrukcije (od 0 do 3) i vrednosti dva registra opšte namene. U tu svrhu, preuzmite datoteke *top\_shell.vhd*, *cpu\_clk\_gen.vhd*, *lcd\_clk\_gen.vhd* i *lcd\_driver.vhd* i uvežite ih u projekat.

Dopuniti vrh hijerarhije (*top.vhd*) sa izlazima iz Tabele 3-1. Ovo su izlazi koje koristi LCD handler kako bi ispisao potrebne informacije na LCD ekranu. Nakon ovoga, ukoliko ste koristili imena prolaza kao što smo ih definisali na vežbama, dovoljno je samo povezati prolaze na pinove FPGA i pokrenuti implementaciju. Povezivanje sa pinovima izvršiti prema Tabeli 3-2.

Tabela 3-1. Prolazi registra

|  |  |  |
| --- | --- | --- |
| **Prolaz(i)** | **Smer** | **Funkcija** |
| oPC [15:0] | out | vrednost programskog brojača |
| oIR [13:0] | out | trenutno izvršavana instrukcija |
| oPHASE [1:0] | out | faza izvršenja instrukcije |
| oR0 – oR7 [15:0] | out | vrednosti registara R0 – R7 |

Tabela 3-2. Povezivanje prolaza na komponente TLL5000 platforme

|  |  |  |
| --- | --- | --- |
| **Port** | **Smer** | **TLL5000** |
| iCLK | in | CLK |
| inRST | in | RESET |
| iPB | in | JOY2 |
| iSW [7:0] | in | SW7 – SW0 |
| oLCD\_D[3:0] | out | LCD\_D [3:0] |
| oLCD\_EN | out | LCD\_EN |
| oLCD\_RW | out | LCD\_R/#W |
| oLCD\_RS | out | LCD\_RS |

Pritiskom na taster JOY2 daje se takt procesoru. Svakim pritiskom izvrpšavanje instrukcije se propagira za jednu fazu. Nekada se jednim pritiskom može poslati više od jednog takta procesoru, zbog neidealnosti mehanike prekidača, ali to ne predstavlja problem. Stanja u nekim od značajnih registara se mogu pročitati na LCD ekranu, i to:

* A – programski brojač,
* I – instrukcioni registar,
* P – faza izvršavanja instrukcije,
* R0 – registar opšte namene R0.

Pored registra R0, na ekranu se ispisuje i vrednost još jednog registra opšte namene, a koji je to registar, zavisi od stanja prekidača. Ukoliko je uključen prekidač SW**X** (gde **X** može biti od 1 do 7), biće prikazana vrednost registra R**X**. Ukoliko je uključeno više prekidača, najveći prioritet ima registar sa najmanjim rednim brojem.

Projektovanje procesora – proširenje skupa instrukcija

# Programiranje CPU

U instrukcijskoj memoriji napisati program koji:

* Sa memorijskih lokacija na adresama 0x0000, 0x0001 i 0x0002 čita tri operanda i smešta ih u registre R0, R1 i R2 respektivno. Memorijske lokacije treba da imaju neku nenultu početnu vrednost.
* Računa sledeće:
  + R3 <- (R0 + R1) <<< 2 -- aritmetičko pomeranje
  + R4 <- R2 >>> 3
  + R4 <- R4 + R3
* RezultatizregistraR4smešta u memoriju na lokaciju na adresi 0x0003.

Dopuniti program tako da se izvršavanje navedenih operacija izvršava u petlji 4 puta:

* U prvoj iteraciji čitaju se 0x0000, 0x0001 i 0x0002, a rezultat upisuje u 0x0003,
* U drugoj iteraciji čitaju se 0x0001, 0x0002 i 0x0003, a rezultat upisuje u 0x0004,
* U trećoj iteraciji čitaju se 0x0002, 0x0003 i 0x0004, a rezultat upisuje u 0x0005,
* U poslednjoj iteraciji čitaju se 0x0003, 0x0004 i 0x0005, a rezultat upisuje u 0x0006.