Projektovanje procesora – upravljačka jedinica

# Osnovni skup instrukcija

Zadatak ove vežbe je realizacija upravljačke jedinice procesora sa Slike 1-1 za izvršavanje proizvoljnog programa, za razliku od upravljačke jedinice iz dodatnog zadatka prethodne vežbe koja izvršava samo jedan program. U nastavku sledi željeni opis rada procesora.



Slika 1‑1. Arhitektura procesora

Trenutna instrukcija iINSTR[14:0] se smešta u registar instrukcije (IR), ako je aktivan signal dozvole sIR\_WE.

Ulazni podaci se preko 16-bitnog ulaza iDATAi multipleksera upisuju u registre A i B. Izlaz multipleksera su16-bitne magistrala podataka sMUXA i sMUXB, kojesu ujedno i ulazi registara A i B. Aritmetičko logička jedinica izvršava operacije tako što se prvi operand smesti u registar A i drugi operand smesti u registar B u jednom ciklusu takta, a u narednom ciklusu serezultat operacijesmešta u registar C, registar za smeštanje rezultata, a nakon toga rezultat se može proslediti u neki od registara opšte namene R0-R7.

Upravljačka jedinica treba da generiše selekcione i signale dozvole ostalih delova digitalnog sistema kao sto su: sREG\_WE[7:0] (dozvole upisa u registre opšte namene R0-R7), sA\_WE, sB\_WE i sC\_WE (dozvole upisa u registre A, B i C), sIR\_WE (dozvola upisa u instrukcioni registar), sPC\_IN, sPC\_EN i sPC\_LOAD (kontrolni signali i ulaz programskog brojača), sMUXA\_SEL[3:0] i sMUXB\_SEL[3:0] (odabir prvog i drugog operanda), sALU\_SEL (kod operacije aritmetičko logičke jedinice) i sMEM\_WE (dozvola upisa u memoriju).

Na izlaz procesora se prosleđuju:16-bitni izlazni podatak ka memoriji (oDATA), 16-bitna vrednost programskog brojača (oPC) i dozvola upisa u memoriju (oMEM\_WE).

Procesor za početak treba da obavlja operacije koje prikazuje Tabela 1.1. U pitanju su operacije koje direktno izračunava aritmetičko-logička jedinica. U levoj koloni tabele su navedena imena instrukcija i oznake operanada, a u desnoj koloni funkcije pojedinih instrukcija. Sintaksa RZ←[RX] označava da se sadržaj registra X upisuje u registar Z.

Tabela 1-1. Instrukcije procesora

|  |  |  |
| --- | --- | --- |
| **Operacija** | **Funkcija** | **Kod instrukcije** |
| mov Rz,Rx | RZ ← [RX] | 000000 |
| add Rz,Rx,Ry | RZ ← [RX] + [RY] | 000001 |
| sub Rz,Rx,Ry | RZ ← [RX] - [RY] | 000010 |
| and Rz,Rx,Ry | RZ ← [RX] & [RY] | 000011 |
| or Rz,Rx,Ry | RZ ← [RX] | [RY] | 000100 |
| not Rz,Rx | RZ ← not [RX] | 000101 |
| inc Rz,Rx | RZ ← [RX] + 1 | 000110 |
| dec Rz,Rx | RZ ← [RX] - 1 | 000111 |
| shl Rz,Rx | RZ ← shl [RX] | 001000 |
| shr Rz,Rx | RZ ← shr [RX] | 001001 |
| ashl Rz,Rx | RZ ← ashl [RX] | 001010 |
| ashr Rz,Rx | RZ ← ashr [RX] | 001011 |

Svaka instrukcija je kodovana sa 15 bita. Lista instrukcija se smešta u posebnu, instrukcionu memoriju, odvojenu od memorije za smeštanje podataka. Format instrukcije na bitskom nivou je IIIIIIZZZXXXYYY. Pri tome je sa IIIIII označen kod intrukcije, sa XXX adresa registra RX, sa YYY adresa registra RY, a sa ZZZ adresa registra Z.

Iako su za navedene instrukcije dovoljna tri bita, ovde se rezerviše šest iz razloga proširenja skupa instrukcija koji će se raditi u ovoj i narednoj vežbi.

Izvršenje svih instrukcija je raspoređeno u četiri faze (četiri periode takta):

1. prihvat insntrukcije (*Instruction Fetch)*
2. dekodovanje instrukcije*(Instruction Decode)*,
3. izvršenje instrukcije *(Execute)* i
4. pamćenje rezultata (*Write Back)*.

Ovakvo raspoređivanje izvršenja instrukcija je potrebno zbog arhitekture procesora, koja sadrži četiri registarska stepena:

* u prvoj fazi se vrednost smešta u registar IR,
* u drugoj fazi se, na osnovu trenutne instrukcije u IR, vrednost smešta u registre A i B,
* u trećoj fazi se, na osnovu vrednosti u IR, A i B, vrši računanje ivrednost smešta u registar C,
* u četvrtoj fazi se rezultat računanja iz registra C smešta u registre R0-R7.

Upravljačka jedinica predstavlja automat koji konstantno kruži između četiri stanja koji predstavljaju prethodno navedene četiri faze izvršenja instrukcije. Funkcija prelaza tog automata je jednostavna, automat bezuslovno prelazi iz stanja FETCH u stanje DECODE, iz stanja DECODE u stanje EXECUTE, iz stanja EXECUTE u stanje WRITE\_BACK i iz stanja WRITE\_BACK u stanje FETCH.

Funkcija izlaza, odn. vrednosti kontrolnih signala za ostatak procesora, definiše se na osnovu trenutnog stanja automata i trenutno izvršavane instrukcije koja se nalazi u registru IR i predstavlja ulaz upravljačke jedinice (iINSTR). U tabeli 1-2 sledi spisak upravljačkih signala čiju vrednost treba postaviti u datoj fazi izvršenja instrukcije. U fazama u kojima signal nije naveden, on treba da bude **neaktivan**.

Tabela 1-2.Aktivni upravljački signali po fazama izvršenja instrukcije

|  |  |  |  |
| --- | --- | --- | --- |
| **Instruction Fetch** | **Instruction Decode** | **Execute** | **Write-Back** |
| sIR\_WE | sMUXA\_SEL  sMUXB\_SEL  sA\_WE  sB\_WE | sALU\_SEL  sC\_WE | sREG\_WE  sPC\_EN |

U prvoj fazi se, signalom sIR\_WE omogućuje prihvatanje instrukcije u registar IR.

U drugoj fazi se vrši odabir prvog i drugog operanda signalima sMUXA\_SEL i sMUXB\_SEL, a odabrani operandi upisuju u registre A i B postavljanjem signala dozvole sA\_WE i sB\_WE na 1.

U trećoj fazi se odabira operacija aritmetičko-logičke jedinice i vrednost upisuje u registar C, kao i statusne flip-flopove.

U poslednjoj fazi se rezultat operacije upisuje u registar opšte namene (samo jedan bit u signalu sREG\_WE treba da bude aktivan) i programskom brojaču se, pomoću signala sPC\_EN, dozvoljava uvećavanje adrese za 1, kako bi se u narednom taktu preuzela naredna instrukcija.

Za sada ne koristimo sledeće upravljačke signale: sPC\_IN, sPC\_LOAD, sMEM\_WE.

Zadaci:

1. Napisati tablicu kontrolnih signala za naveden osnovni skup instrukcija, definišući vrednost svakog kontrolnog izlaza (izlaza upravljačke jedinice) u svakoj fazi izvršavanjainstrukcije.
2. Realizovati upravljačku jedinicu u VHDL jeziku definišući:
   1. Proces koji definiše automat upravljačke jedinice
   2. Funkciju prelaza (zajedno sa ili odvojeno od prethodnog procesa, po želji)
   3. Funkciju izlaza kao poseban kombinacioni proces

Projektovanje procesora – upravljačka jedinica

# Bezuslovni skok

Dopuniti skup instrukcija instrukcijom bezuslovnog skoka iz Tabele 2-1. Ova instrukcija ima sledeći format: IIIIIAAAAAAAAA. Instrukcija se sastoji iz 2 dela:

* iINSTR (13 downto 9) – kod instrukcije,
* iINSTR (8 downto 0) – adresa na koju se skače.

Tabela 2-1. Instrukcija bezuslovnog skoka

|  |  |  |
| --- | --- | --- |
| **Operacija** | **Funkcija** | **Kod instrukcije** |
| jmp ADDR | PC ← ADDR | 010000 |

Adresa na koju se skače treba u fazi WRITE\_BACK da se upiše u programski brojač. U tu svrhu se koriste signali sPC\_IN i sPC\_LOAD. Pošto se rezultat operacije ne upisuje u registre opšte namene, novi format instrukcije ne smeta ostatku sistema, te stoga nije potrebno menjati generisanje kontrolnih signala. Potrebno je samo promeniti generisanje signala oREG\_WE u WRITE\_BACK fazi, kao i signala oC\_WE u EXECUTE fazi tako da se tokom instrukcije JMP vrednost ne upisuje ni u jedan registar, kao ni u registre rezultata i statusnih bita.

Program u instrukcijskoj memoriji dopuniti instrukcijom skoka na početak programa i u simulaciji proveriti da li se program ponovo izvršava.

# Uslovni skok

Dopuniti skup instrukcija instrukcijama uslovnog skoka iz Tabele 2-2. Ove instrukcije imaju isti format kao instrukcija bezuslovnog skoka, sa drugačijim kodom instrukcije. Statusni biti ZERO, SIGN i CARRY so označeni sa Z, S i C respektivno.

Tabela 2-1. Instrukcije uslovnog skoka

|  |  |  |
| --- | --- | --- |
| **Operacija** | **Funkcija** | **Kod instrukcije** |
| jmpz ADDR | if Z=1 PC ← ADDR | 010001 |
| jmps ADDR | if S=1 PC ← ADDR | 010010 |
| jmpc ADDR | if C=1 PC ← ADDR | 010011 |
| jmpnz ADDR | if Z=0 PC ← ADDR | 010101 |
| jmpns ADDR | if S=0 PC ← ADDR | 010110 |
| jmpnc ADDR | if C=0 PC ← ADDR | 010111 |

Kao i kod instrukcije bezuslovnog skoka, potrebno je samo promeniti generisanje signala oREG\_WE u WRITE\_BACK fazi, kao i signala oC\_WE u EXECUTE fazi tako da se tokom instrukcije JMP vrednost ne upisuje ni u jedan registar, kao ni u registre rezultata i statusnih bita.

# Vrh hijerarhije računarskog sistema CPU + memorija

U datoteci *top.vhd* realizovati vrh hijerarhije sistema, u kome je procesor povezan sa memorijama, prikazan na Slici 3-1. Za sada definisati samo jedan izlaz vrha hijerarhije, oDATA [15:0] i na njega povezati signal sDATA\_ST [15:0]. Ovaj izlaz je neophodan kako bi alat izvršio sintezu dizajna (dizajn mora imati makar jedan izlaz da bi imalo smisla vršiti njegovu implementaciju).



Slika 3‑1. Vrh hijerarhije sistema