9. Vaja: mikroprocesor (2)

Univerza v Ljubljani
Fakulteta za elektrotehniko

Model mikroprocesorja bomo nadgradili z vhodno izhodno enoto in tremi ukazi:

JZE A – skok na naslov A, če je v akumulatorju vrednost 0

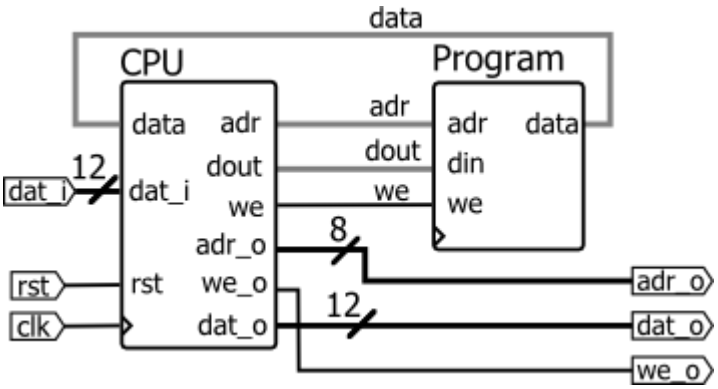
INP A – naloži v akumulator vrednost iz DAT_I na naslovu A

OUTP A – prenesi akumulator v DAT_O ob naslovu A

Simulator mikroprocesorja: cpu.html

Naloga

Dodaj v Proc.vhd zunanje priključke **(port)** in povezave med komponentami, kot prikazuje shema:



Nadgradnja CPU

a. Naredi pogojni skok: procesor naj izvede skok JZE, kadar je vrednost v akumulatorju enaka 0, sicer naj skoka ne izvede. Poglej, kako je opisan skočni ukaz JMP in na podoben način naredi še JZE !

b. Dodaj zunanje priključke: 12-bitni nepredznačeni vhod **dat_i** in , 8-bitni izhod **adr_o**, 12-bitni izhod **dat_o** in enobitni izhod **we_o**.

   - V stanju *zajemi* naj se na **adr_o** prenese spodnjih 8-bitov iz podatkovnega vodila **data**, če predstavljajo zgornji 4 biti ukaz INP ali OUTP.
     Kadar je v tem stanju na vodilu ukaz OUTP, naj se vrednost akumulatorja prepiše v izhodni register **dat_o.**

   - V stanju *izvedi* naj se podatek iz vodila **dat_i** prenese v akumulator (podobno kot ukaz LDA). Signal **we_o** dobi vrednost '1' ob ukazu OUTP in v stanju *izvedi*, v vseh ostalih primerih naj ima **we_o** vrednost '0'.

c. Naredi simulacijo s testnim programom:

```
type memory is array(0 to 255) of unsigned(11
downto 0);
signal m : memory := (
0=> inp & x"00",
1=> jze & x"00",
2=> lda & x"07",
3=> add & x"08",
4=> sta & x"07",
5=> outp & x"01",
6=> jmp & x"00",
7=> x"007",
8=> x"003",
others => x"000" );
```

Program v simulatorju:

cpu.html

```
start:  inp vh
        jze start
        lda a
        add b
        sta a
        outp vh
        jmp start
vh      di 0
a       db 7
b       db 3
```

Predlagane razširitve nabora ukazov:

```
-- razširjen nabor ukazov za samostojno delo
 constant jcs: koda := "0110";  -- skok, ce je prenos (carry=1)

 constant nota: koda := "0000"; -- logicna negacija
 constant anda: koda := "1100"; -- logicna IN
 constant ora:  koda := "1101"; -- logicna ALI

 constant shl: koda := "1110"; -- pomik akum za eno v levo (mnozenje z 2)
 constant shr: koda := "1111"; -- pomik akum za eno v desno (deljenje z 2)
```