8. Vaja: mikroprocesor

Centralno procesno enoto iz prejšnje vaje bomo povezali s pomnilnikom (Program) in naredili preprost mikroprocesor. Ukazi CPU bodo definirani v knjižnici, tako da bo vsebina programskega pomnilnika podobna zbirni kodi (assembler), ki je najnižji programski jezik za programiranje mikroprocesorjev. Ukazi v zbirniku:

LDA M – *naloži vrednost iz pomnilniške lokacije M v akumulator*

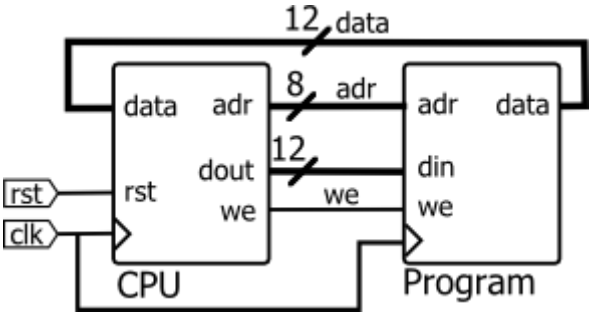
JMP A – *skok na novo pomnilniško lokacijo z naslovom A*

ADD M – *prištej k akumulatorju vrednost iz pomnilniške lokacije M*

Model CPU bomo nadgradili z ukazom za shranjevanje podatkov:

STA M – shrani akumulator v *pomnilnik na lokacijo M*

12-bitni mikroprocesor s pomnilnikom



Mikroprocesor bomo zgradili iz komponent in lastne knjižnice. V opisu mikroprocesorja in programa bomo uporabljali ukazne kode v obliki simboličnih konstant, ki so definirane v knjižnici procpak.vhd.

```
subtype  koda is unsigned(3 downto 0);
constant lda: koda := "0001"; -- nalozi iz pomnilnika v akumulator
constant jmp: koda := "0100"; -- skok na novo pomnilnisko lokacijo
constant add: koda := "1000"; -- pristej vrednost iz pomnilnika
```

Knjižnico uporabimo tako, da jo vključimo na začetek opisa vezja s stavkom:

```
library work;
use work.procpak.all;
```

Pomnilnik

V datoteki program.vhd je opis pomnilnika vrste ROM v katerem je kratek program. Pomnilnik vsebuje 12-bitne mikroprocesorske ukaze in podatke. Ukaz je sestavljen iz 4-bitne kode, ki jo predstavljajo zgornji štiri biti, spodnjih osem bitov pa je naslov v pomnilniku. V pomnilnik smo zapisali kratek program iz treh ukazov: najprej naložimo v akumulator vrednost (005) iz pomnilniške lokacije 03, nato prištejemo isto vrednost, tretji ukaz pa je brezpogojni skok na drugi naslov (01).

```
type memory is array(0 to 255) of unsigned(11 downto 0);
```

```
signal ram : memory := (
 lda & x"03",
 add & x"03",
 jmp & x"01",
 x"005",
 others => x"000" );
```

Naloga

a. Naloži datoteke procpak.vhd, proc.vhd in program.vhd in popravi **cpu.vhd** tako, da bo uporabljala simbolične konstante (lda, add in jmp) iz knjižnice.

b. Med priključke mikroprocesorja dodaj 12-bitni izhod **dout**, ki naj bo enak akumulatorju, ter enobitni izhodni signal **we**. Za ta signal napiši kombinacijsko logiko:
   - **we** dobi vrednost '1', kadar smo v stanju *zajemi* in je na podatkovnem vhodu ukaz sta: **data**(11:8) = sta
   - sicer naj ima **we** vrednost '0'

c. Popravi model pomnilnika, tako da bo omogočal shranjevanje podatkov. Uporabi vhodna signala **din** in **we.**
   - Dodaj logiko za pisanje v sinhroni proces. Kadar je signal **we='1',** se v pomnilnik **ram** na naslovu **adr** vpiše vhodni podatek iz **din**.

d. Dodaj v program ukaz **sta** in preizkusi delovanje na simulatorju,

```
type memory is array(0 to 255) of unsigned(11 downto 0);

signal ram : memory := (
 lda & x"04",
 add & x"04",
 sta & x"04",
 jmp & x"01",
 x"005",
 others => x"000" );
```