7. vaja: Mikroprocesor

1 Programski pomnilnik

Mikroprocesor bere ukaze in podatke iz programskega pomnilnika (prog). Naredi model pomnilnika, ki vsebuje kratek program z 8 besedami. Pomnilnik naj ima 16-biten vhod **din**, 8-biten vhod **adr**, enobiten kontrolni vhod **wr** in 16-biten izhod **data**.

adr	data	pomen
0	x"1005"	ukaz LDA 05, naloži v akumulator podatek iz naslova 05 (a=10)
1	x"8006"	ADD 06, prištej akumulatorju podatek iz naslova 06 (a=a+4)
2	x"8007"	ADD 07, prištej iz naslova 07 (a=a+1)
3	x"2005"	STA 05, shrani akumulator na naslov 05 (ram(05)=a)
4	x"4002"	ukaz JMP 02, skoči na naslov 02
5	x"000A"	vrednost 10
6	x"0004"	vrednost 4
7	x"0001"	vrednost 1

a) Deklariraj polje z vsebino pomnilnika in opiši branje pomnilnika iz naslova **adr**.

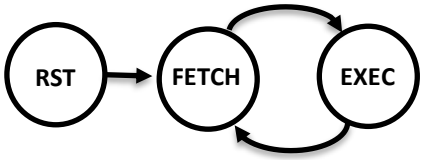
```
rom: 256u16 = x"1005",x"8006",x"8007",x"2005",x"4002",10,4,1;
data = rom(adr);
```

b) Dodaj stavek, ki ob **wr**=1 zapiše vrednost iz **din** v pomnilnik na naslov **adr**. Preizkusi delovanje na simulaciji in shrani prevedeno datoteko prog.vhd.

2 Krmilna enota mikroprocesorja

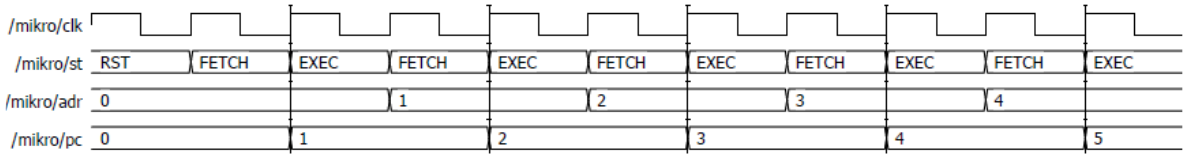
a) Naredi ogrodje opisa mikroprocesorja (mikro.vhd) z zunanjimi priključki:
- clk in reset,
- 16-bitna vhodna vrata pin,
- 16-bitna izhodna vrata pout.

b) Deklariraj notranje signale: 8-bitna **adr** in **pc** (programski števec), 16-bitna **din** in **data**. Deklariraj tudi spremenljivko za register stanj s tremi stanji: RST, FETCH, EXEC, ki se spreminjajo po diagramu:



c) Vključi v opis programki pomnilnik in ga poveži z notranjimi signali.

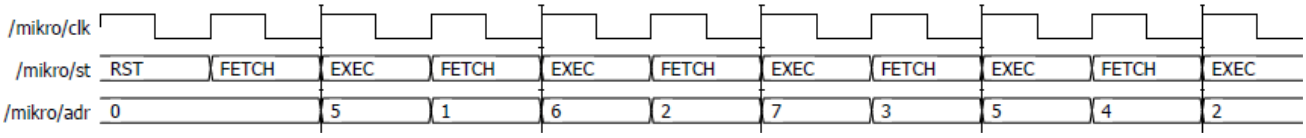
d) Opiši spreminjanje naslovnega signala in programskega števca ter preveri delovanje na simulaciji
- programski števec **pc** je ob resetu 0, v stanju EXEC pa se povečuje za 1
- naslov **adr** naj bo v stanju FETCH enak programskemu števcu



3 Zajem parametrov ukaza

Mikroprocesor dobi v stanju FETCH iz pomnilnika ukaz, ki ga sestavljata ukazna koda in parameter, npr. ukaz LDA 5 ima vrednost parametra 5. Parameter učnega procesorja predstavlja pomnilniški naslov, kjer mikroprocesor dobi podatek za izvebo ukaza.

a) Deklariraj 4-bitni signal za ukazno kodo in 12-bitni signal za ukazni parameter. Ukazna koda naj bo enaka zgornjim štirim bitom podatka iz pomnilnika, parameter pa spodnjim 12 bitom.
b) Dopolni logiko za določanje naslova: v staju EXEC naj bo naslov enak parametru.



4 Dekodiranje in izvedba ukazov LDA in ADD

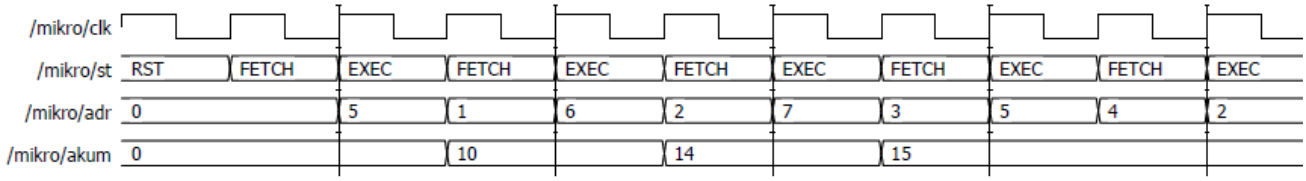
Ukaz LDA naloži podatek iz pomnilnika v akumulator mikroprocesorja, ukaz ADD pa podatek prišteje k trenutni vrednosti akumulatorja. Ukazne kode so v pomnilniku zapisane s številkami (strojna koda). Za lažje obvladovanje modela procesorja, bomo namesto številskih kod deklarirali konstante v novi VHDL datoteki.

a) Naredi VHDL paket (mpack.vhdl) v katerem sta deklarirani dve 4-bitni konstanti: LDA in ADD

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.numeric_std.all;

package mpack is
 constant LDA: unsigned(3 downto 0) := "0001";
 constant ADD: unsigned(3 downto 0) := "1000";
```

b) Vključi paket v opis mikroprocesorja (use work.mpack.all) in deklariraj še 16-biten signal akum.
c) Zapiši stavke, ki bodo v stanju EXEC izvedli ukaza LDA in ADD*

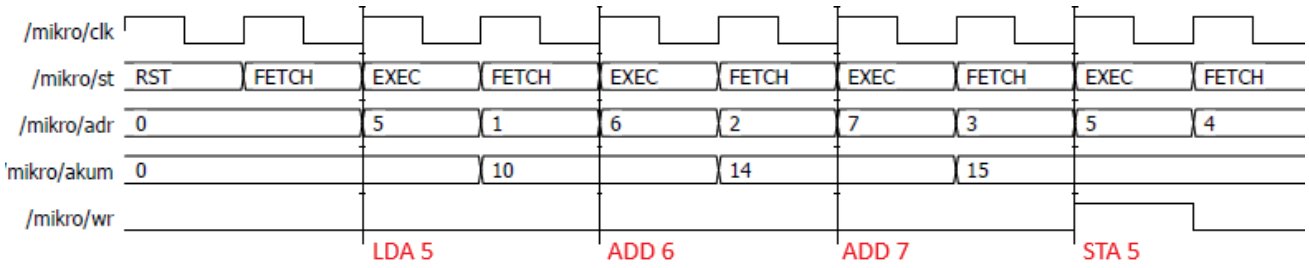


* Za rešitev te naloge bo potrebno dodati še kakšen register. Ukazna koda je namreč na voljo v stanju FETCH, informacijo o zadnji ukazni kodi pa potrebujemo tudi v stanju EXEC.

5 Zapis vrednosti v pomnilnik

Ukaz STA shrani vrednost iz akumulatorja v pomnilnik na naslov, ki ga določa parameter.

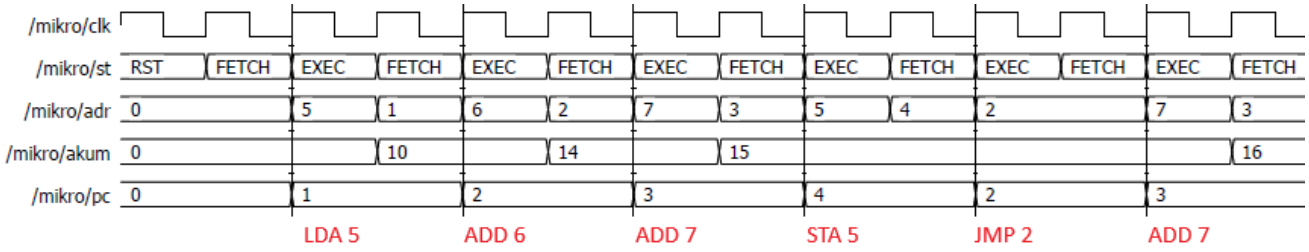
a) Dodaj v paket novo konstanto z ukazom STA, ki ima vrednost 0010.
b) V opisu mikroprocesorja deklariraj enobitni signal **wr** in ga poveži na pomnilnik. Na podatkovni vhod pomnilnika **din** naj bo vezan akumulator **akum**.
c) Napiši logiko, ki aktivira signal **wr** v stanju EXEC ob izvajanju ukaza STA. Signal mora biti na 1 natančno v tem ciklu, sicer vpis ne bo pravilno deloval – preveri na časovnem diagramu in na simulaciji, kjer opazuj tudi vsebino pomnilnika!



6 Dekodiranje skočnega ukaza

Skočni ukaz JMP povzroči, da program nadaljuje izvajanje na naslovu določenem s parametrom ukaza. Ukaz nastavi novo vrednost programskega števca **pc**.

a) Dodaj v paket novo konstanto s skočnim ukazom JMP (ukazna koda 0100) in v opis mikroprocesorja stavke, ki izvedejo skok.



7 Vhodno-izhodni ukazi

Mikroprocesor povežemo v sistem preko vhodno-izhodnih vrat. Ukaz INP prenese vrednost iz vhoda **pin** v akumulator, ukaz OUTP pa vsebino akumulatorja prenese na izhodna vrata **pout**.

a) Dodaj v paket novo konstanto z ukazom OUTP in v opis dodaj v opis mikroprocesorja logiko za izhodni register **pout**.

b) V pomnilniku spremeni program tako, da bo namesto stavka STA 5 stavek OUTP 0. Uporabi spletni prevajalnik za CPE 16, ki prevede zbirniški program v strojno kodo.

```
        lda x
        add y
z:      add 1
        outp 0
        jmp z
x       db 10
y       db 4
```

c) Preveri delovanje s simulacijo.