7. vaja: Mikroprocesor

Naredili bomo 12-bitni mikroprocesor, ki zna izvajati tri ukaze: LDA, ADD in JMP.

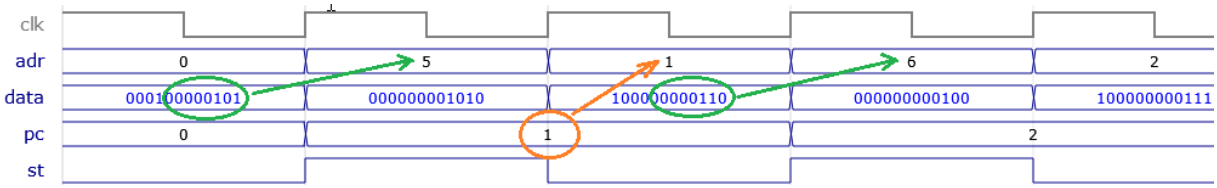
Osnovni model mikroprocesorja

Opiši mikroprocesor s programskim pomnilnikom in notranjimi signali: 8-bitni naslov (**adr**) in 12-bitni podatek (**data**). V vezje naj bo vključen model pomnilnika s testnim programom:

adr	data	Pomen
0	x"105"	ukaz LDA 05, naloži v akumulator podatek iz naslova 05 (a=10)
1	x"806"	ADD 06, prištej akumulatorju podatek iz naslova 06 (a=a+4)
2	x"807"	ADD 07, prištej iz naslova 07 (a=a+1)
3	x"205"	STA 05, shrani akumulator na naslov 05 (ram(05)=a)
4	x"402"	ukaz JMP 02, skoči na naslov 02
5	x"00A"	vrednost 10
6	x"004"	vrednost 4
7	x"001"	vrednost 1

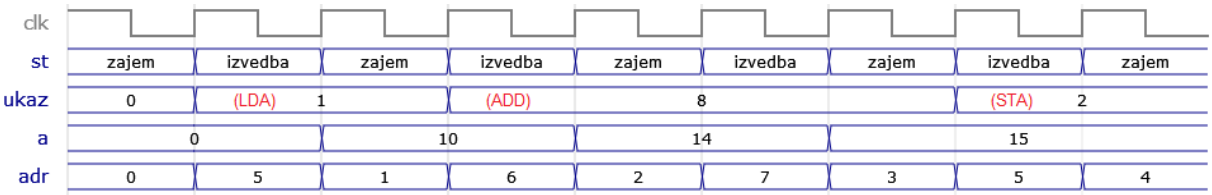
```
rom: 16u12 = x"105",x"806",x"807",x"205",x"402",x"00A",x"004",x"001";
data=rom(adr);
```

1. Deklariraj 8-bitne notranje signale: programski števec **pc**, trenutni naslov **adr** in naslednji naslov ter signal za stanje procesorja (zajem ukaza ali izvedba ukaza). Opiši logiko za določanje stanja, naslova in programskega števca.

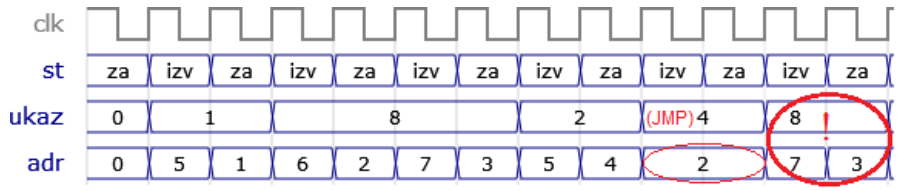


- stanje se ciklično izmenjuje (0 in 1 oz. *zajem* in *izvedba*)
- naslov pomnilnika **adr** izmenično določata programski števec **pc** in strojni ukaz
- ko dobimo ukaz, shranimo spodnjih 8 bitov, ki predstavljajo naslov v naslednjem ciklu
- ob koncu stanja *zajem* povečamo programski števec za 1, da bomo naslednjič brali nov ukaz

2. Dodaj v vezje 12-bitni akumulator (**a**) in 4-bitni ukazni register (**ukaz**). Ukazni register shrani vrednost zgornjih štirih bitov podatka v stanju *zajem*. Opiši delovanje ukazov LDA in ADD. V stanju izvajanja naloži LDA podatek v akumulator, ADD pa podatek prišteje akumulatorju:



3. Dodaj še opis skočnega ukaza JMP (koda "0100"). Izvedba ukaza JMP naj povzroči, da se bo naslednji ukaz zajemal na naslovu, ki ga določajo spodnji biti ukaza JMP.



Namig: ob izvedbi JMP je naslov **adr** že pravi, povzročiti moramo, da bo tudi ob naslednjem ciklu **adr** ostal enak.