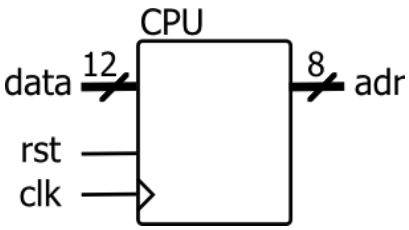
7. Vaja: CPU

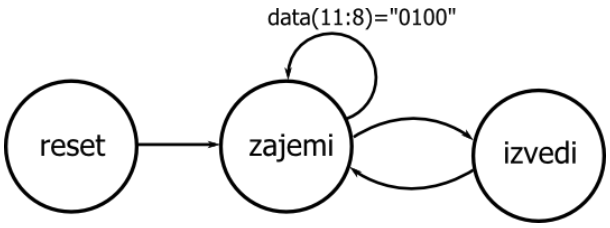
Naredili bomo model 12-bitne centralno procesne enote (CPU) z akumulatorjem, ki zna izvajati preproste strojne ukaze iz programskega pomnilnika. Strojni ukazi so 12-bitne kombinacje: zgornji 4 biti določajo vrsto ukaza, spodnjih 8 pa določa pomnilniški naslov. Za začetek bomo naredili enoto s tremi strojnimi ukazi:

"0001nnnnnnnn": naloži vrednost s pomnilniške lokacije nnnnnnnn v akumulator
"0100nnnnnnnn": skoči na naslov nnnnnnnn
"1000nnnnnnnn": prištej k akumulatorju vrednost iz pomnilniške lokacije nnnnnnnn

12-bitna centralno procesna enota



Začetni model centralno procesne enote naj ima na vhodu uro in reset ter 12-bitno podatkovno vodilo **data,** na izhodu pa 8-bitno naslovno vodilo **adr**, na katerega bo kasneje priključen pomnilnik s programom. Enota začne po resetu zajemati in izvajati ukaze iz pomnilnika. Korake izvajanja ukazov krmili sekvenčni stroj (avtomat) s tremi stanji:



V krmilnem delu vezja se nahajata še dva 8-bitna registra: programski števec **(pc)** in ukazni naslov (**inst_adr**), ki naj bosta deklarirana kot signala tipa unsigned z začetno vrednostjo 0.

a. Naredi novo vezje cpu.vhd z definicijami zunanjih priključkov in notranjimi signali. Deklariraj tudi nov podatkovni tip in notranji signal **st** za sekvenčni stroj:

```
type stanje is (reset, zajemi, izvedi);
signal st: stanje;
```

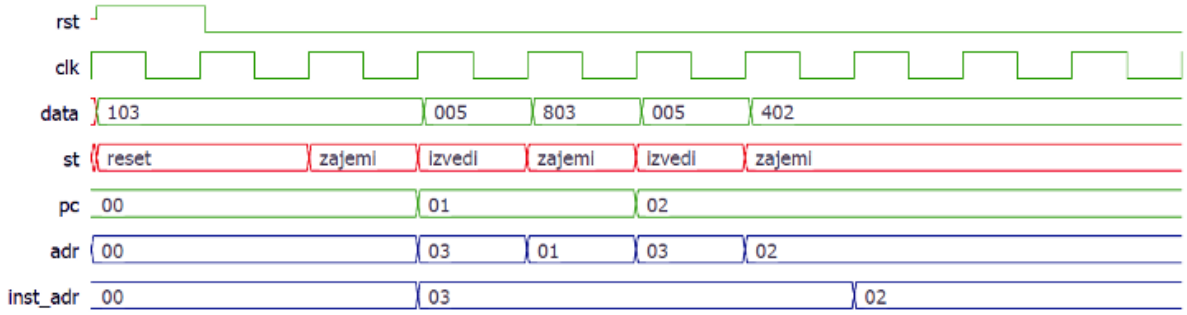
b. Naredi sinhroni proces, v katerem se izmenično izmenjujeta stanji *zajemi* in *izvedi*. Opiši prehajanje stanj avtomata s stavkom case v katerega dodaj še prireditvene stavke za oba

kontrolna registra:

- v stanju reset naj se programski števec (**pc**) postavi na 0
- v stanju zajemi naj se programski števec poveča za 1, razen ob ukazu "0100", ko naj se postavi na vrednost data(7 downto 0)
- v stanju zajemi na se shrani ukazni naslov: inst_adr <= data(7 downto 0);

c. Opiši še izhodno kombinacijko logiko za signal **adr** (izhod se mora nastavljati asinhrono, brez ure):
   - v stanju reset naj bo 0
   - v stanju zajemi naj bo enak programskemu števcu (**pc**)
   - v stanju izvedi naj bo enak ukaznemu naslovu (**inst_adr**)

d. Preizkusi delovanje krmilnega dela CPU na simulaciji s testno strukturo testcpu.vhd, ki ima nastavljene vhode, kot prikazuje diagram:



Podatkovni del CPU

Podatkovna pot procesorja vsebuje akumulator **akum** v katerega se shranjujejo rezultati operacij in register **inst_code** za ukazno kodo.

a. Deklariraj nove notranje signale za registre tipa unsigned: 4-bitni ukazni register **inst_code** in 12-bitni akumulator **akum**

b. V stanju *zajemi* naj se zgornji 4 biti iz vodila **data** shranijo v register **inst_code.** Glede na binarno kombinacijo v tem registru se v stanju *izvedi* izvedejo operacije, ki shranijo rezultat v akumulator:
   - "0001": akum <= data
   - "1000": akum <= akum + data

c. Ponovno naredi simulacijo s testno strukturo in opazuj vrednost akumulatorja. Po izvedbi ukaza s kodo "103" se bo postavil na 5, po izvedbi naslednjega ukaza pa povečal na 10 (šestnajstiško X"00A).