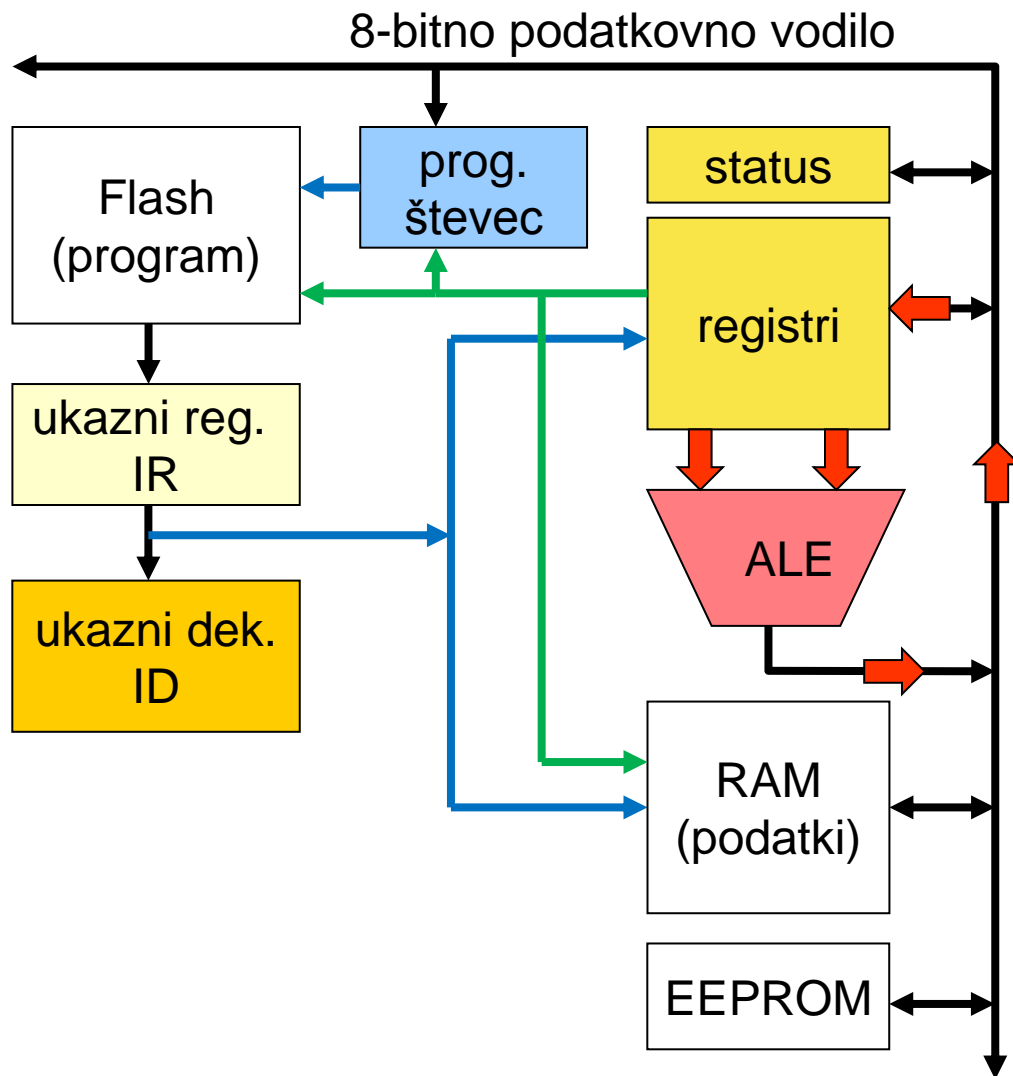
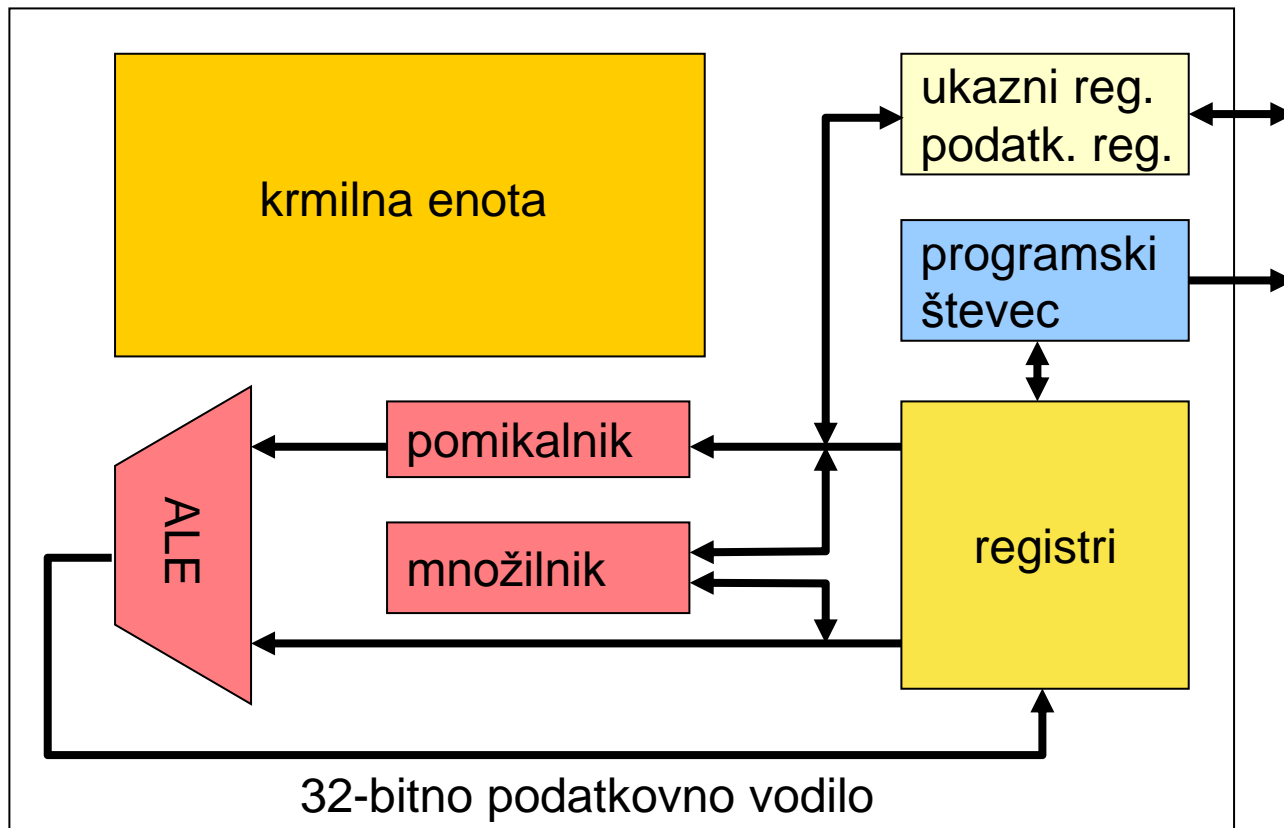


# Centralna procesna enota AVR



# Centralna procesna enota ARM-7



- ▶ Zmogljivi procesorji vsebujejo dodatne računske enote
  - ▶ ciklični pomikalnik, množilnik, enote za plavajočo vejico

# Procesorji v programirljivih vezjih

---

- ▶ izvedba s program. logiko (Soft Core)
  - ▶ vnaprej pripravljene komponente intelektualne lastnine
  - ▶ procesor prevedemo (sinteza, tehnološka preslikava)
  - ▶ zgradbo procesorja in število jeder prilagodimo aplikaciji
  - ▶ Altera Nios (32-bit), LNIV CPE (12-bit)
  
- ▶ izvedba z namenskim vezjem (Hard Core)
  - ▶ programirljivo vezje in mikroprocesor na istem čipu
  - ▶ Bolj zmogljiva / manj prilagodljiva izvedba
  - ▶ ARM-9 (Altera in Xilinx)

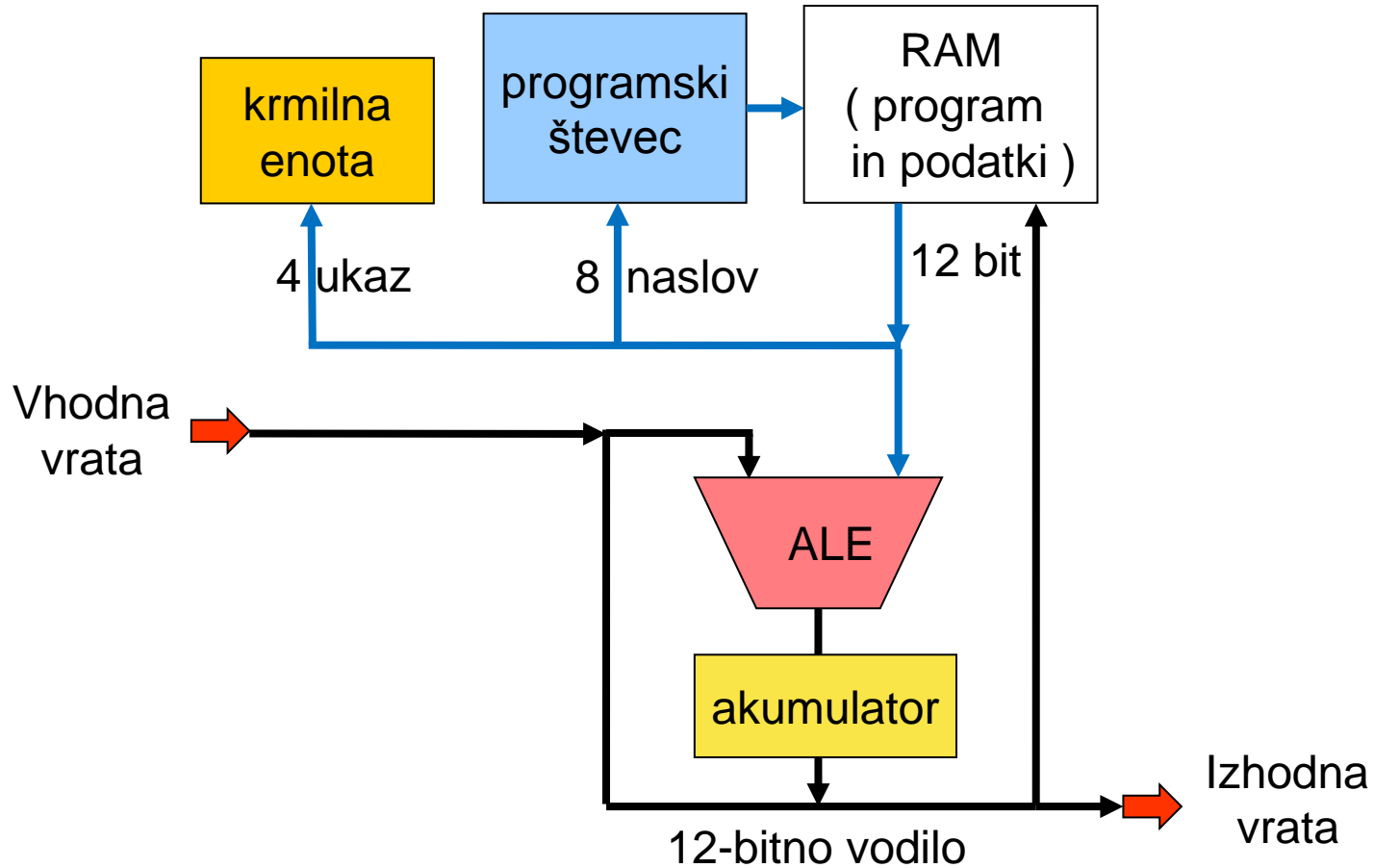
# Prednosti procesorjev v tehnologiji FPGA

---

- ▶ **Sodobna vezja FPGA so dovolj velika za CPE**
  - ▶ zapletena opravila izvajamo na procesorju
- ▶ **Na FPGA lahko naredimo celoten sistem**
  - ▶ manj integriranih vezij v napravi
  - ▶ nismo odvisni od proizvajalcev procesorjev
- ▶ **Prilagodljivost**
  - ▶ CPE popolnoma prilagodimo aplikaciji

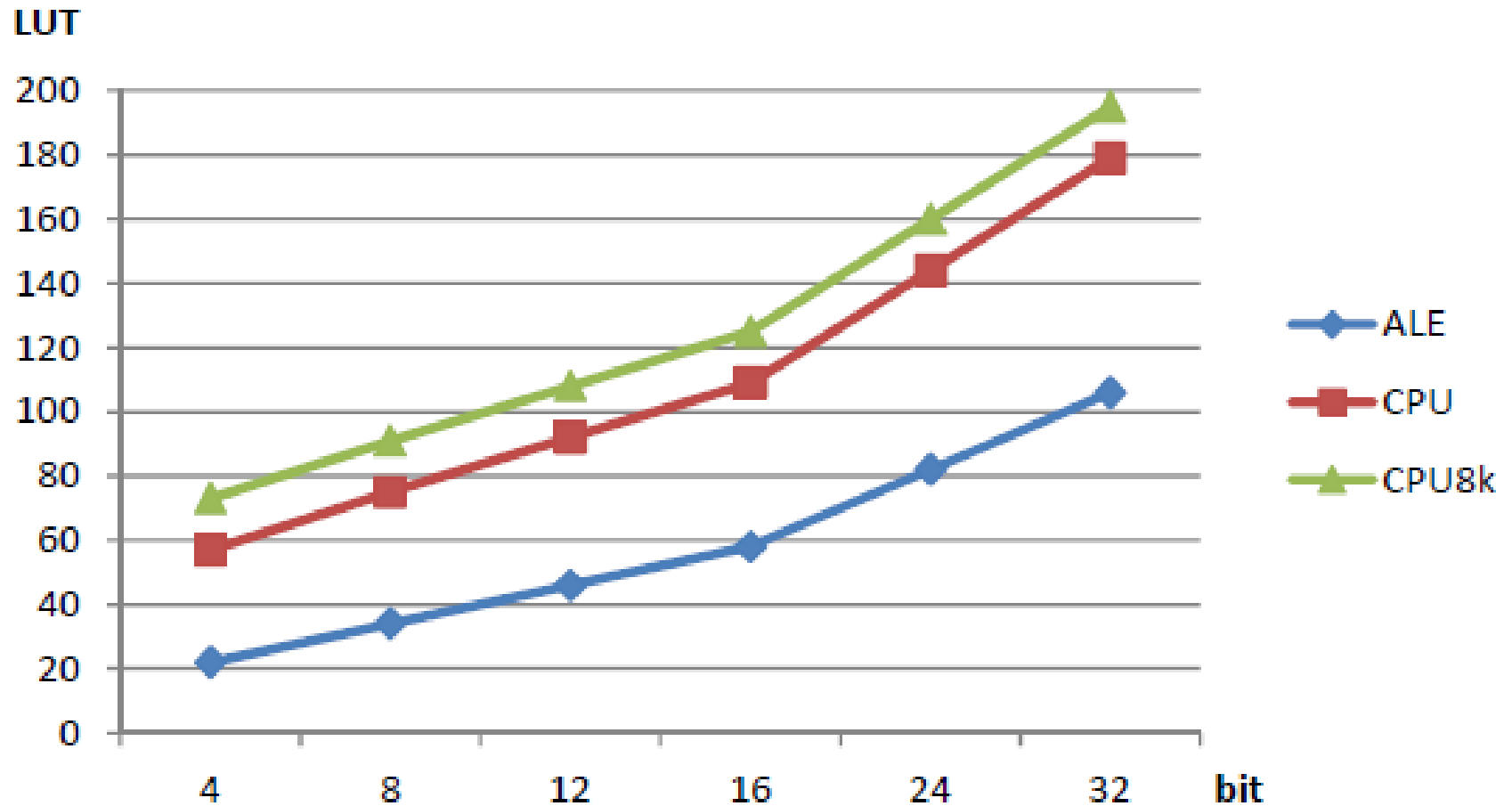
# Enostaven mikroprocesor za FPGA

- ▶ N-bitni procesor, npr. N=12



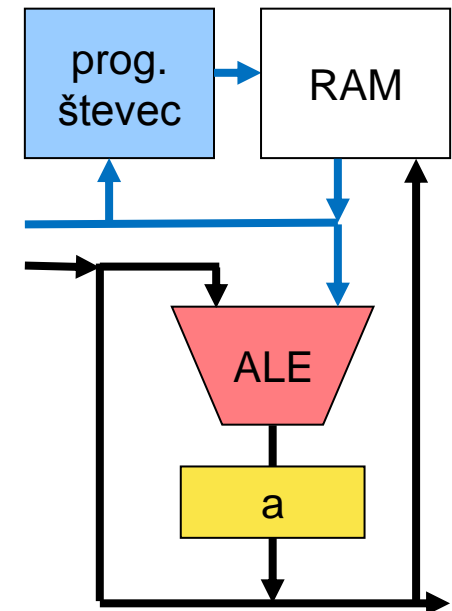
# Processor zasede malo celic v vezju FPGA

---



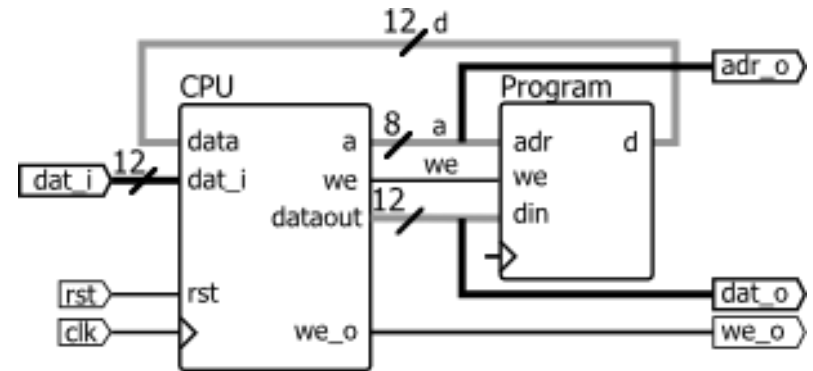
# Osnovni nabor ukazov

<b>lda</b>	koda := "0001"; -- naloži iz RAM $a = [M]$
<b>sta</b>	koda := "0010"; -- shrani v RAM $[M] = a$
<b>add</b>	koda := "0100"; -- prištej $a = a + [M]$
<b>sub</b>	koda := "0101"; -- odštej $a = a - [M]$
<b>anda</b>	koda := "0110"; -- logična $a = a \text{ and } [M]$
<b>ora</b>	koda := "0111"; -- logična $a = a \text{ or } [M]$
<b>jmp</b>	koda := "1000"; -- skok na nov naslov
<b>jze</b>	koda := "1001"; -- skok, če je $a=0$
<b>inp</b>	koda := "0011"; -- naloži iz vhoda $a = IN$
<b>outp</b>	koda := "0111"; -- shrani na izhod $OUT = a$



# Priključki mikroprocesorja

signal	število bitov	opis
clk	1	Sistemska ura.
rst	1	Reset signal.
a	8	Naslov pomnilnika.
data	12	Podatkovni vhod.
dataout	12	Podatkovni izhod.
we	1	Pisanje v pomnilnik.
dat_i	12	Vhodna vrata.
dat_o	12	Izhodna vrata.
we_o	1	Pisanje v izhodna vrata.
adr_o	1	Vhodno-izhodni naslov.





# Programiranje v zbirniku

---

- ▶ Zbirnik je vez med programskimi jeziki (npr. C) in strojno kodo, ki je odvisna od arhitekture
- ▶ Opis majhne in učinkovite programske kode za
  - ▶ nizkonivojske rutine (dostop do perifernih enot)
  - ▶ računsko zahtevna opravila (matematika, grafika)
- ▶ Pri velikih projektih je delo v zbirniku težko, nefleksibilno, prevajalniki jezika C pa so dobro optimizirani

oznaka                      ukaz    operand                      komentar

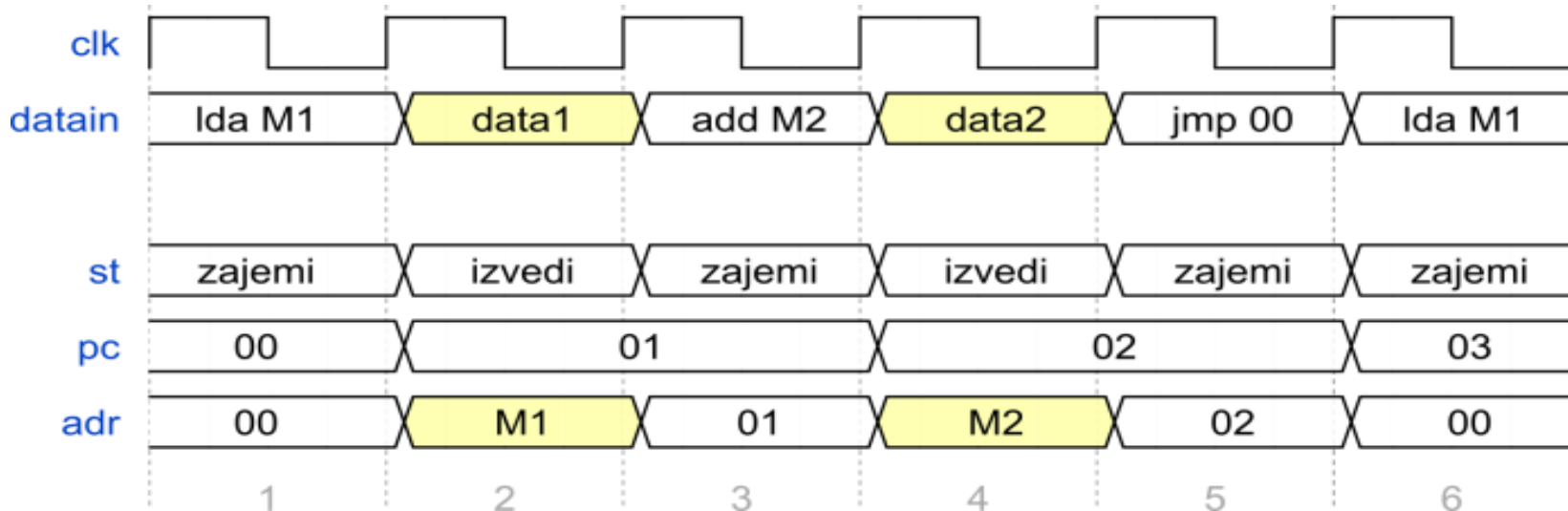
   ↓                      ↙    ↙

start: lda 30    ; naloži konstanto v akumulator

# Zbirnik

- ▶ program, ki računa:  $2+2+2+2+\dots$

```
start:  lda vred      ; naloži vred      0: 0001 0000 0011
        add vred      ; prištej vred     1: 0100 0000 0011
        jmp start     ; skok nazaj      2: 1000 0000 0000
vred    db    2       ; direktiva       3: 0000 0000 0010
```



Simulator: <http://lniv.fe.uni-lj.si/cpu.html>

# LNIV irtuaLAB

Home

## Mikroprocesor CPU 12

Ukazi: NOTA, LDA, STA, INP, JMP, JZE, JCS, OUTP, ADD, SBT, CALL, RET, ANDA, ORA, SHL, SHR

Direktive: DB, DI, DO

Zbirni jezik:

```
start:  inp  vh      ;beri  vh
        jze  start   ;skoči nazaj dokler je vh=0
        add vsota ;prištej k vsoti in shrani
        sta vsota
        outp izh   ;vsoto na izhod
        jmp start  ;skok na začetek

vh     di 0
vsota  db 0
izh   di 1
```

Vhodi

vh	1

Simulacija

PC: 02

Akum: 01

Izhodi

IZH = 1  
IZH = 2  
IZH = 3

Vsebina pomnilnika:

@00	300	500	806	206	701	400	003	000
@08	000	000	000	000	000	000	000	000
@10	000	000	000	000	000	000	000	000
@18	000	000	000	000	000	000	000	000
@20	000	000	000	000	000	000	000	000
@28	000	000	000	000	000	000	000	000
@30	000	000	000	000	000	000	000	000
@38	000	000	000	000	000	000	000	000
@40	000	000	000	000	000	000	000	000
@48	000	000	000	000	000	000	000	000
@50	000	000	000	000	000	000	000	000
@58	000	000	000	000	000	000	000	000
@60	000	000	000	000	000	000	000	000
@68	000	000	000	000	000	000	000	000
@70	000	000	000	000	000	000	000	000
@78	000	000	000	000	000	000	000	000
@80	000	000	000	000	000	000	000	000
@88	000	000	000	000	000	000	000	000
@90	000	000	000	000	000	000	000	000
@98	000	000	000	000	000	000	000	000
@a0	000	000	000	000	000	000	000	000
@a8	000	000	000	000	000	000	000	000
@b0	000	000	000	000	000	000	000	000
@b8	000	000	000	000	000	000	000	000
@c0	000	000	000	000	000	000	000	000
@c8	000	000	000	000	000	000	000	000
@d0	000	000	000	000	000	000	000	000
@d8	000	000	000	000	000	000	000	000