



Laboratorij za načrtovanje integriranih vezij

Univerza *v Ljubljani*  
Fakulteta *za elektrotehniko*



## Programirljivi Digitalni Sistemi

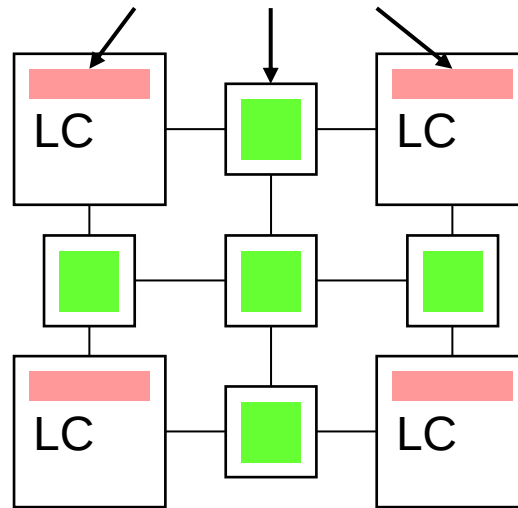
Procesorji

Model računalnika, mikrokontrolnik

# Programirljiva vezja / procesorji

## ▶ programirljivo vezje

- ▶ programski biti določajo strukturo (konfiguracijo)



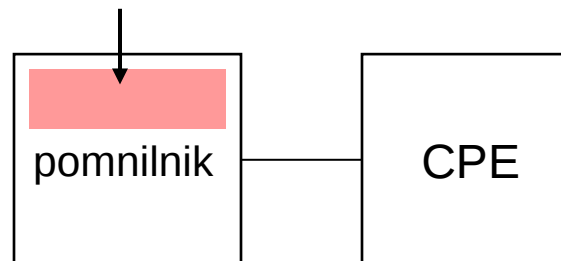
delovanje logičnih celic (LC)

povezave znotraj LC

povezave v povezovalni mreži

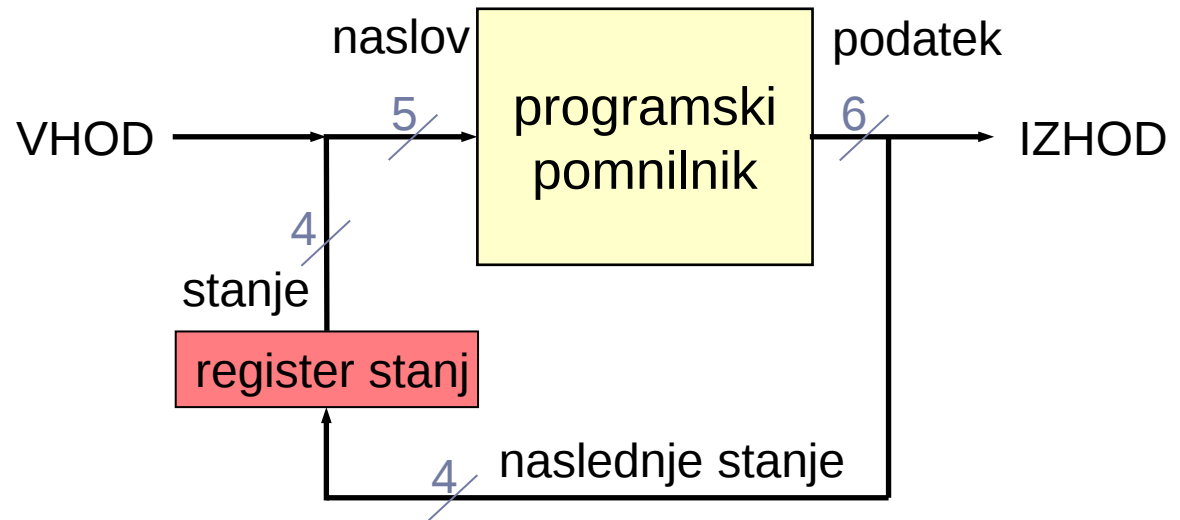
## ▶ mikroprocesor

- ▶ program določa ukaze, ki jih CPE izvaja



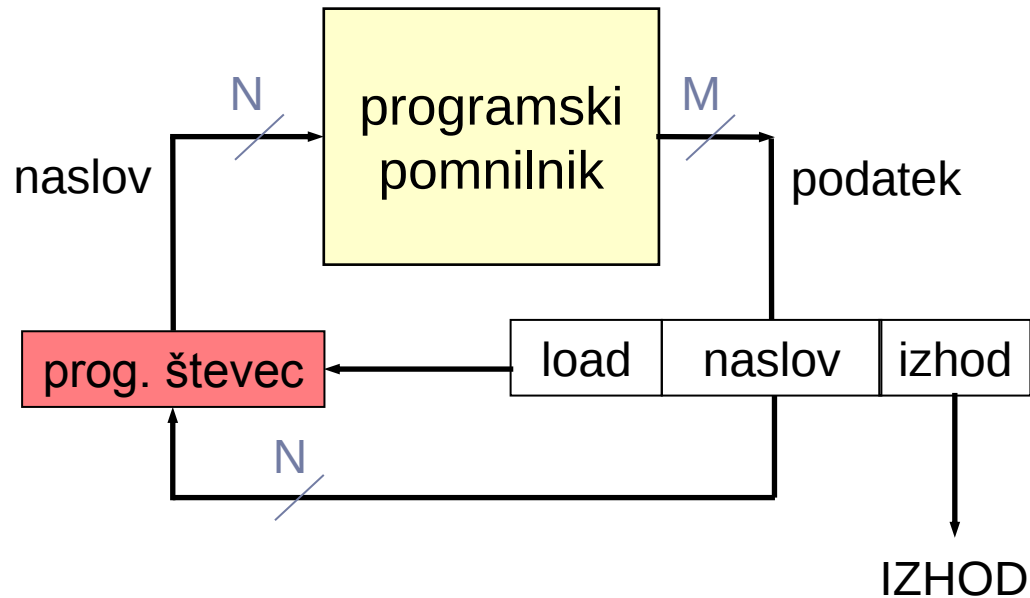
# Mikro-programirano krmilno vezje

- ▶ Krmilni avtomat narejen z LUT je mikro-programiran
  - ▶ *mikro-programirano vezje* ima krmilne vrednosti shranjene v pomnilniku vrste ROM ali RAM
- ▶ *mikroukazi* so besede v pomnilniku
- ▶ *mikroprogram* je zaporedje *mikroukazov*
- ▶ v *programski pomnilnik* vrste RAM lahko vpisujemo *mikroukaze*



# Mikro-programiran krmilnik s števcem

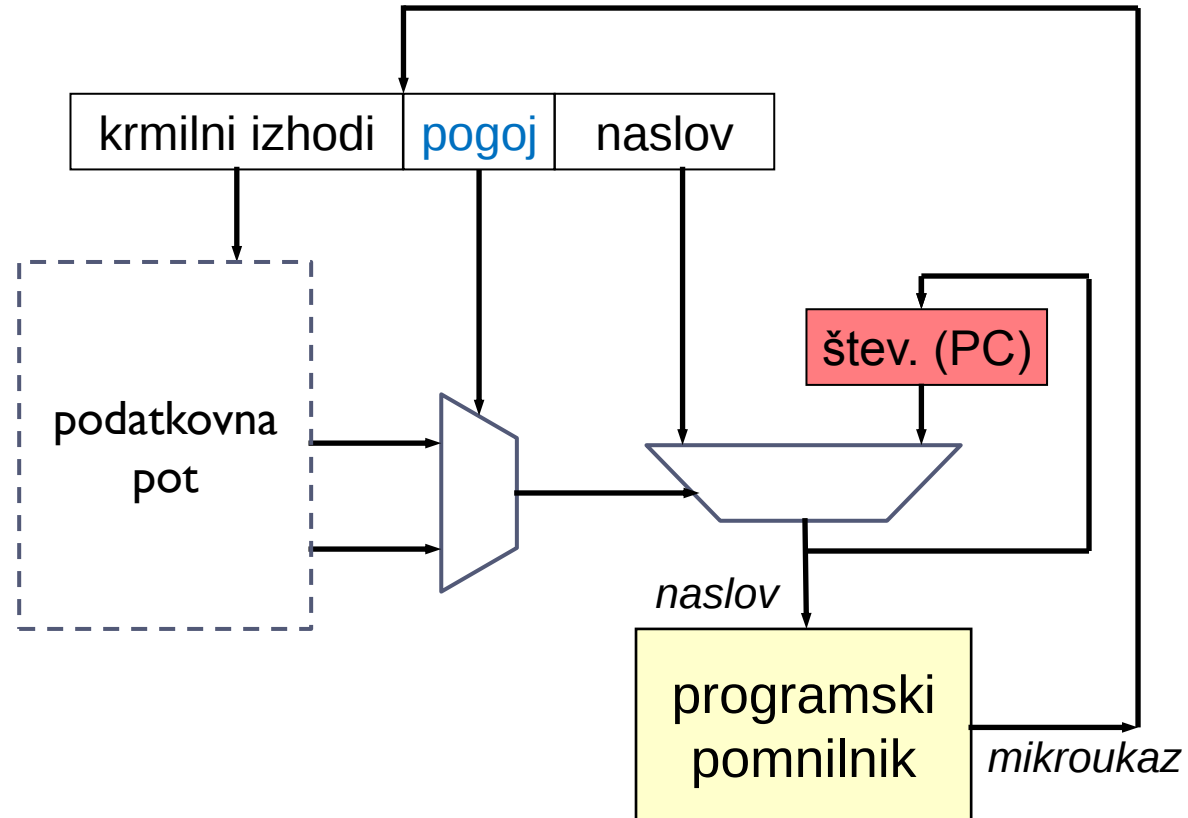
- ▶ Programski števec omogoča izvedbo skokov
  - ▶ izvrševanje algoritmov, ki nimajo vseh mikroukazov v zaporedju



# Mikro-programiran krmilnik s števcem

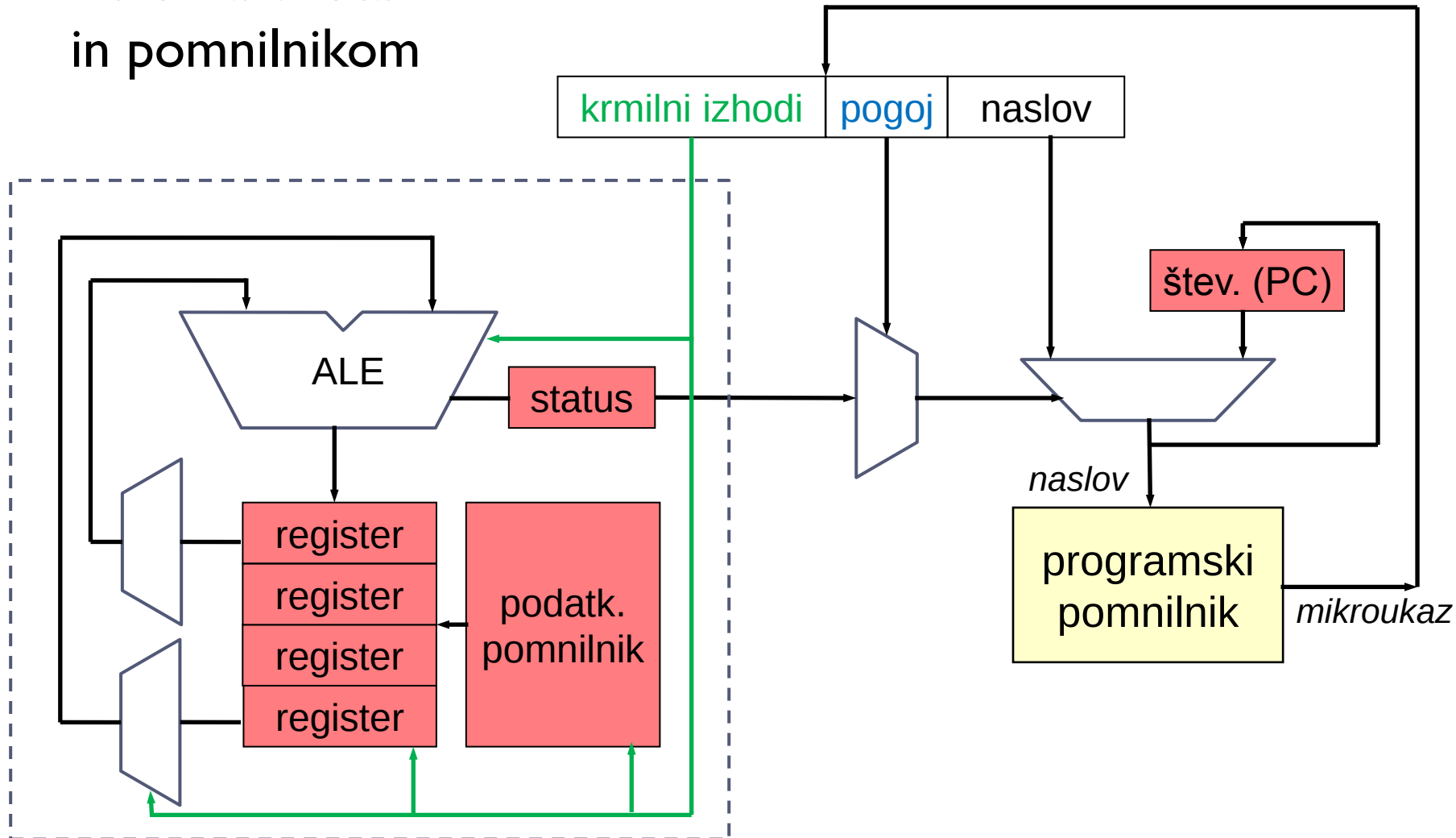
## Mikroukazi

- ▶ mikroprogram se izvaja kot zaporedje mikroukazov
- ▶ mikroukazi določajo krmilne izhode
- ▶ *skočni ukazi* ob pogoju izvršijo skok na nov naslov



# Mikro-programirana krmilna (procesna) enota

- ▶ izvršilna enota z ALE in pomnilnikom



# Mikrooperacije

---

- ▶ Aritmetično-logična enota (ALE) izvaja računske mikrooperacije, npr.:

|                                      |                                              |
|--------------------------------------|----------------------------------------------|
| $r3 \leq r1 + ci$                    | $p1 = 0, p0 = 0, ci = 1$ (povečaj)           |
| $r3 \leq r1 + r2 + ci$               | $p1 = 0, p0 = 1, ci = 1$ (seštej s prenosom) |
| $r3 \leq r1 + (\text{NOT } r2) + ci$ | $p1 = 1, p0 = 0, ci = 1$ (odštej)            |
| $r3 \leq r1 - 1 + ci$                | $p1 = 1, p0 = 1, ci = 1$ (prenesi $r1$ )     |

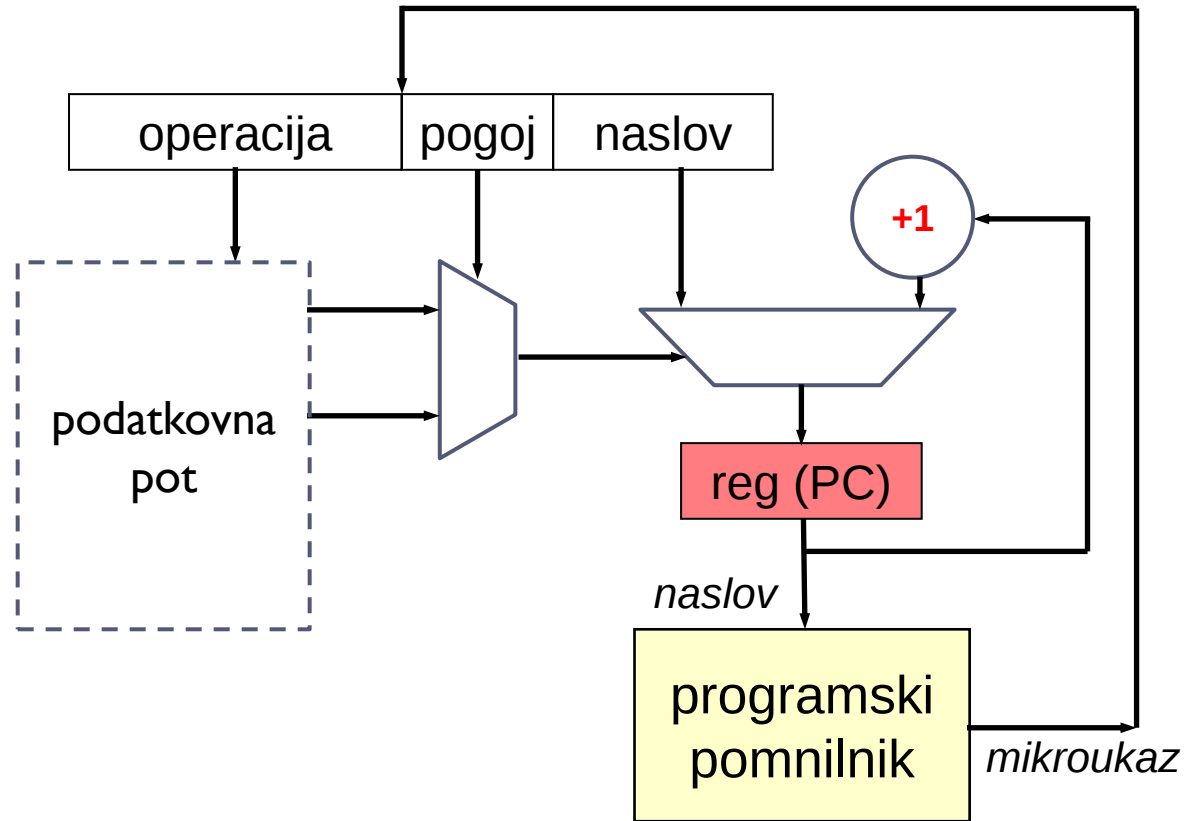
- ▶ Krmilni izhodi izbirajo operacijo, operande in izhodni register
  - ▶ operand je v registru ali v pomnilniku
- ▶ Skočne operacije spremenijo zaporedje izvajanja ukazov
  - ▶ brezpogojni skok, ali pogojni skok, glede na rezultat zadnje operacije – status: prenos, negativni rezultat, rezultat 0...)

# Kode in koraki izvajanja mikrooperacij

---

- ▶ za direktno krmiljenje enot z mikroukazi potrebujemo veliko bitov
  - ▶ npr. 4 bite za ALE, 12 za registre, 16 za pomnilnik....
- ▶ krmilni signali so kodirani
  - ▶ potrebujemo ukazni dekode
- ▶ mikroukazi se izvršijo v več urnih ciklih
  - ▶ bolj optimalna kritična pot in višja frekvenca ure
  - ▶ nekaterih operacij ne moremo narediti v enem ciklu
    - ▶ npr. branje in pisanje v pomnilnik

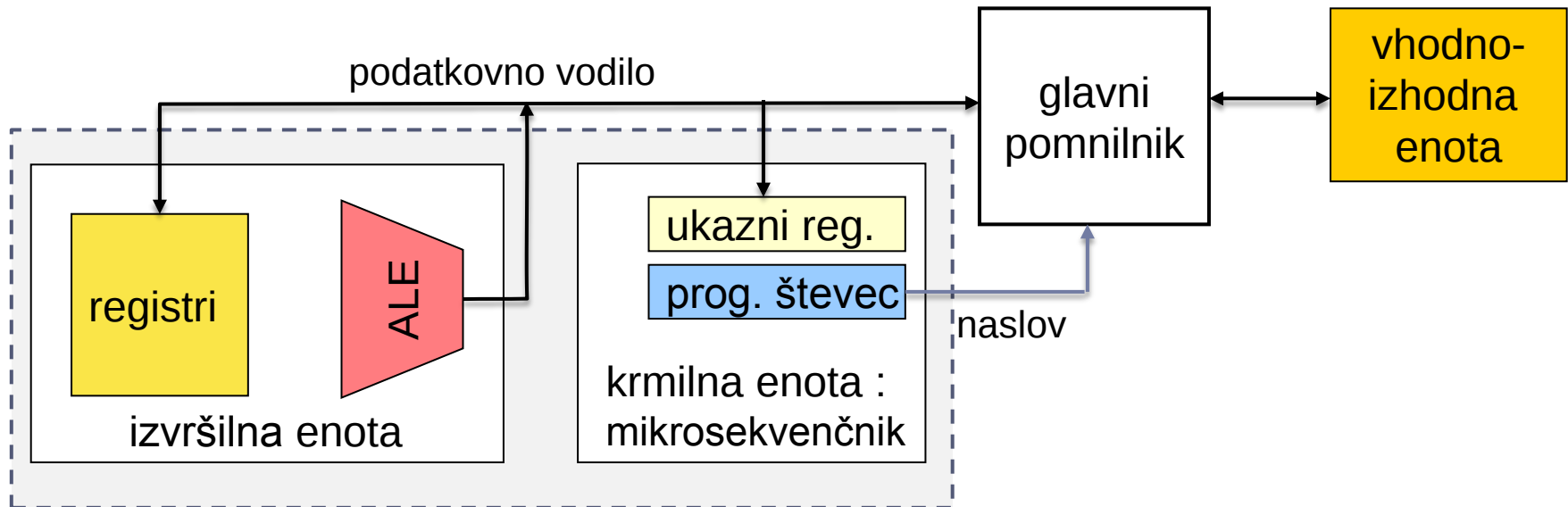
# Mikrosekvenčni in podatkovna pot



- ▶ Podatkovna pot izvaja registrske (mikro)operacije
  - ▶ registri, ALE, podatkovni pomnilnik

# Von Neumannov model računalnika

- ▶ Centralno procesna enota (CPE) in glavni pomnilnik
  - ▶ vhodno – izhodna enota skrbi za komunikacijo z zunanjo svetom



- ▶ delovanje CPE določa nabor ukazov
  - ▶ ukazi so prilagojeni programskemu jeziku (C/C++, Java)

# Osnovni gradniki mikroprocesorja

---

- ▶ Mikroprocesor na integriranem vezju vsebuje izvršilno in krmilno enoto
- ▶ Mikroprocesor potrebuje za delovanje:
  - ▶ zunanjo uro in reset
  - ▶ zunanji pomnilnik s programskimi ukazi in podatki
  - ▶ vhodno in izhodno (**periferno**) enoto za komunikacijo z okolico
    - ▶ periferna enota je lahko del pomnilnika (na določenih naslovih)
    - ▶ ali pa poteka komunikacija preko posebnih V/I ukazov
- ▶ V praksi potrebujemo vsaj dve vrsti pomnilnika
  - ▶ za program takšnega, ki ohranja vsebino (ROM, Flash)
  - ▶ za delovne podatke pa pomnilnik s hitrim branjem in pisanjem (RAM – Random Access Memory)

# Vrste mikroprocesorjev

---

- ▶ Mikroprocesorji v zmogljivih računalnikih – CISC
- ▶ Mikroprocesorji v vgrajenih sistemih – RISC
- ▶ **Complex Instruction Set Computer - CISC**
  - ▶ veliko število ukazov
  - ▶ enostavne mikrooperacije do sestavljenih operacij
  - ▶ čas za izvajanje ukazov je zelo različen
- ▶ **Reduced Instruction Set Computer - RISC**
  - ▶ manjši nabor ukazov
  - ▶ vse operacije so enostavne in učinkovite za izvrševanje
  - ▶ večina ukazov se izvede v enem urnem ciklu

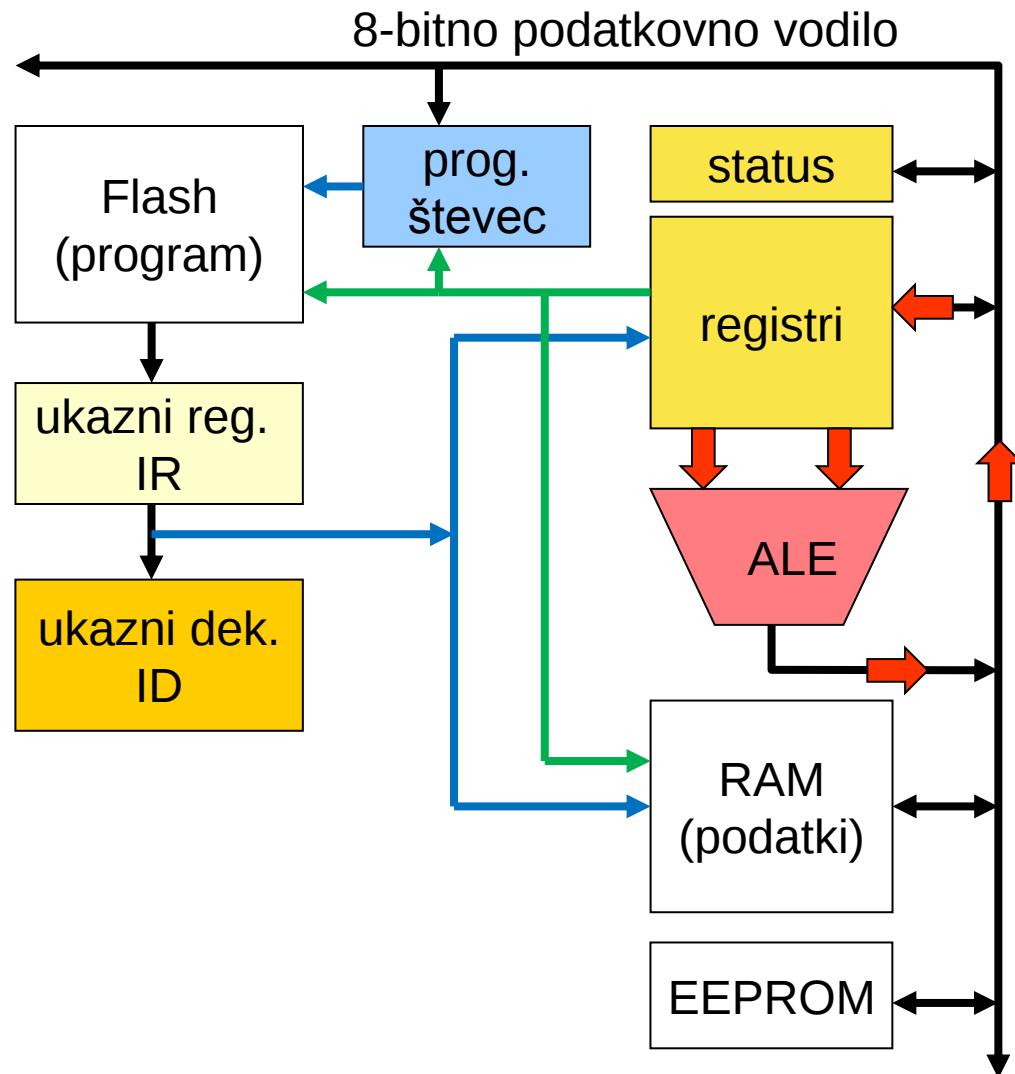
# Mikroprocesorji za vgrajene naprave

---

- ▶ Mikrokrmilniki so sistemi na integriranem vezju, ki vsebujejo
  - ▶ mikroprocesorsko jedro (izvršilno in krmilno enoto),
  - ▶ programski in podatkovni pomnilnik,
  - ▶ ter različne V/I vmesnike:
    - ▶ vzporedna vrata (Port)
    - ▶ zaporedne komunikacijske vmesnike: I2C, SPI, UART
    - ▶ analogno / digitalne (A / D) in D / A pretvornike
    - ▶ modulatorje (PWM), časovnike, števec...
    - ▶ komunikacijske krmilnike: Ethernet MAC, USB
- ▶ Primer mikrokrmilnikov
  - ▶ Atmel AVR, 8-bitni procesor na razvojnem sistemu Arduino
  - ▶ ARM-7, 32-bitni procesor na razvojnem sistemu Š-ARM

# Delovanje CPE mikrokrmilnika AVR

- ▶ programski števec vsebuje naslov naslednjega ukaza
- ▶ IR vsebuje naslednji ukaz
- ▶ ID vsebuje trenutni ukaz
- ▶ Registri: R0-R31
- ▶ ALE – glej označeno interno podatkovno pot
- ▶ Flash programski pomnilnik 16 oz. 32 bitni ukazi
- ▶ RAM vsebuje delovne podatke
- ▶ EEPROM trajni podatki
  - ▶ počasen dostop, omejeno število vpisov v pomnilnik



# Programiranje v zbirniku

---

- ▶ Zbirnik je nizkonivojski programski jezik
- ▶ Mikroukazi so odvisni od arhitekture (različni za različne procesorje)
- ▶ Zbirnik je med jezikom C in strojno kodo
- ▶ Uporaba v manjših vgrajenih sistemih (npr. PIC, AVR)
- ▶ Pri velikih projektih je delo v zbirniku težko, nefleksibilno, prevajalniki jezika C pa so dobro optimizirani
- ▶ Zbirnik se uporablja za:
  - ▶ nizkonivojske rutine
  - ▶ računsko zahtevna opravila (matematika, grafika)
  - ▶ reverzni inženiring

# Ukazi v zbirniku

---

- ▶ Ukazi v zbirniku predstavljajo procesorske mikrooperacije
- ▶ Naloži neposredno konstanto ( $Rd \leq K$ )
- ▶ **LDI Rd, K**
  - ▶ strojna koda: 1110 bbbb rrrr bbbb
    - ▶ omejitve: registri R16-R31; konstanta  $K < 255$
- ▶ **LDI R16, \$2C**
  - ▶ koda: 1110 0010 0000 1100 ali **\$E20C**
- ▶ Seštej in shrani v Rd ( $Rd \leq Rd + Rr$ )
- ▶ **ADD Rd, Rr**
  - ▶ strojna koda: 0000 11rd dddd rrrr
    - ▶ Rd in Rr sta katerakoli registra R0-R31
- ▶ **ADD R16, R17**
  - ▶ ddddd je 10000, rrrrr je 10001, koda ukaza: **\$0F01**

# Strojni program

---

- ▶ Naloži program v pomnilnik

- ▶ na naslov 0 shranimo \$E20C      0000: \$E20C LDI R16, \$2C
- ▶ na naslov 1 shranimo \$E01F      0001: \$E01F LDI R17, \$0F
- ▶ na naslov 2 shranimo \$0F01      0002: \$0F01 ADD R16, R17

- ▶ Izvedemo zaporedje treh ukazov

- ▶ nastavi prog. števec na 0 in izvedi 3 ukazne cikle (prenos/izvedi)

- ▶ Pregled rezultata:

$$\begin{aligned} R16 &= R16 + R17 \\ &= \$2C + \$0F \\ &= \$3B \end{aligned}$$

- ▶ Kaj se bo izvedlo po teh treh ukazih?

- ▶ ne vemo, odvisno kaj je zapisano v pomnilniku Flash...

## Vhodno-izhodne (periferne) enote

- ▶ vzporedna vrata (I/O port) vsebujejo 3 registre

nastavi  
4 vh. in →  
4 izh. bite

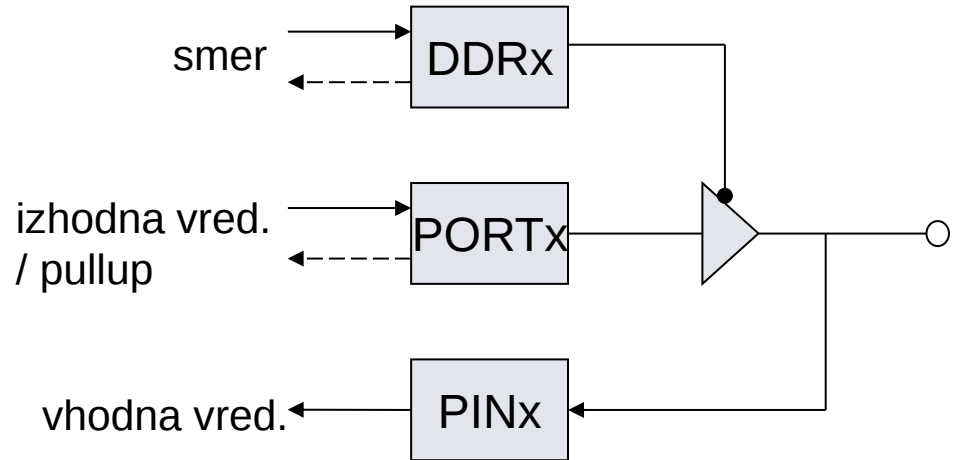
**LDI R19, \$F0**  
**OUT DDRB, R19**

izhodi  $\rightarrow$

**LDI R21, \$50**  
**OUT PORTB, R21**

beri  
vhode →

## IN R20, PORTB



| DDxn | PORTxn | PUD<br>(in SFIOR) | I/O    | Pull-up | Comment                                     |
|------|--------|-------------------|--------|---------|---------------------------------------------|
| 0    | 0      | X                 | Input  | No      | Tri-state (Hi-Z)                            |
| 0    | 1      | 0                 | Input  | Yes     | Pxn will source current if ext. pulled low. |
| 0    | 1      | 1                 | Input  | No      | Tri-state (Hi-Z)                            |
| 1    | 0      | X                 | Output | No      | Output Low (Sink)                           |
| 1    | 1      | X                 | Output | No      | Output High (Source)                        |