Osnove mikroprocesorske elektronike

Marko Jankovec

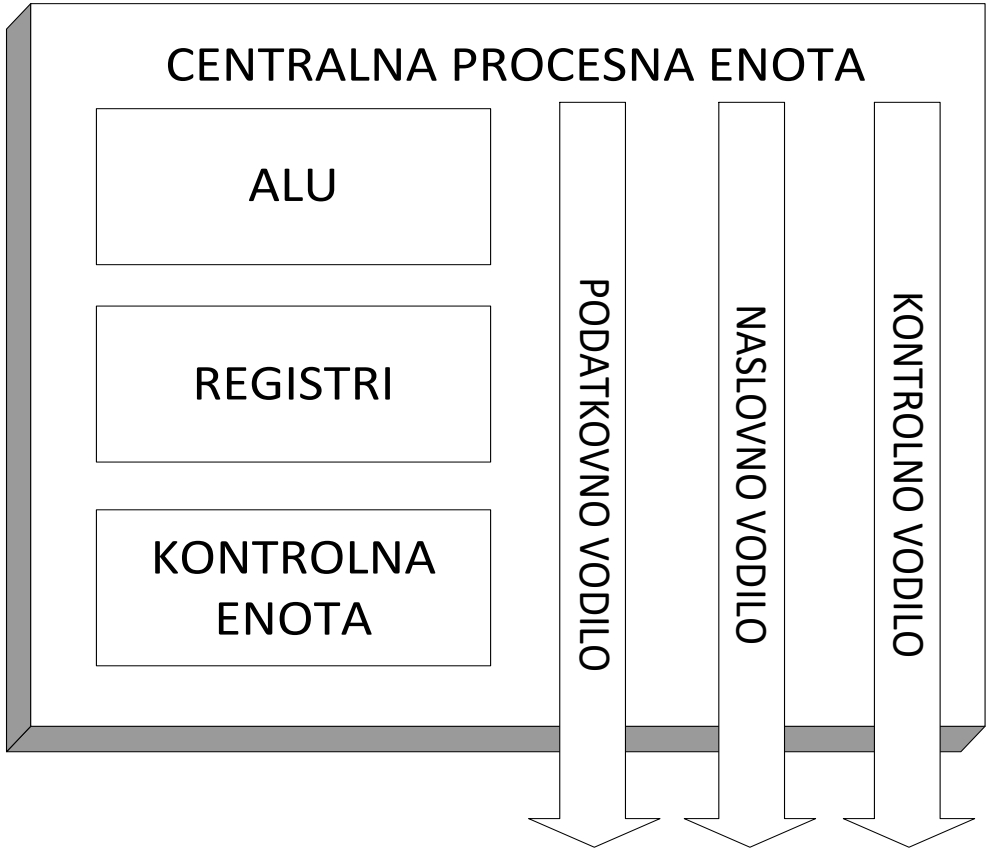
Arhitekture mikroprocesorjev

Arhitektura vs. organizacija

- Arhitektura mikroprocesorja je funkcionalna slika, kot jo vidi načrtovalec programske opreme.
  - nabor ukazov
  - število bitov za predstavitev vrst podatkov
  - načini naslavljanja pomnilnika
  - načini dostopa do vhodno-izhodnih naprav

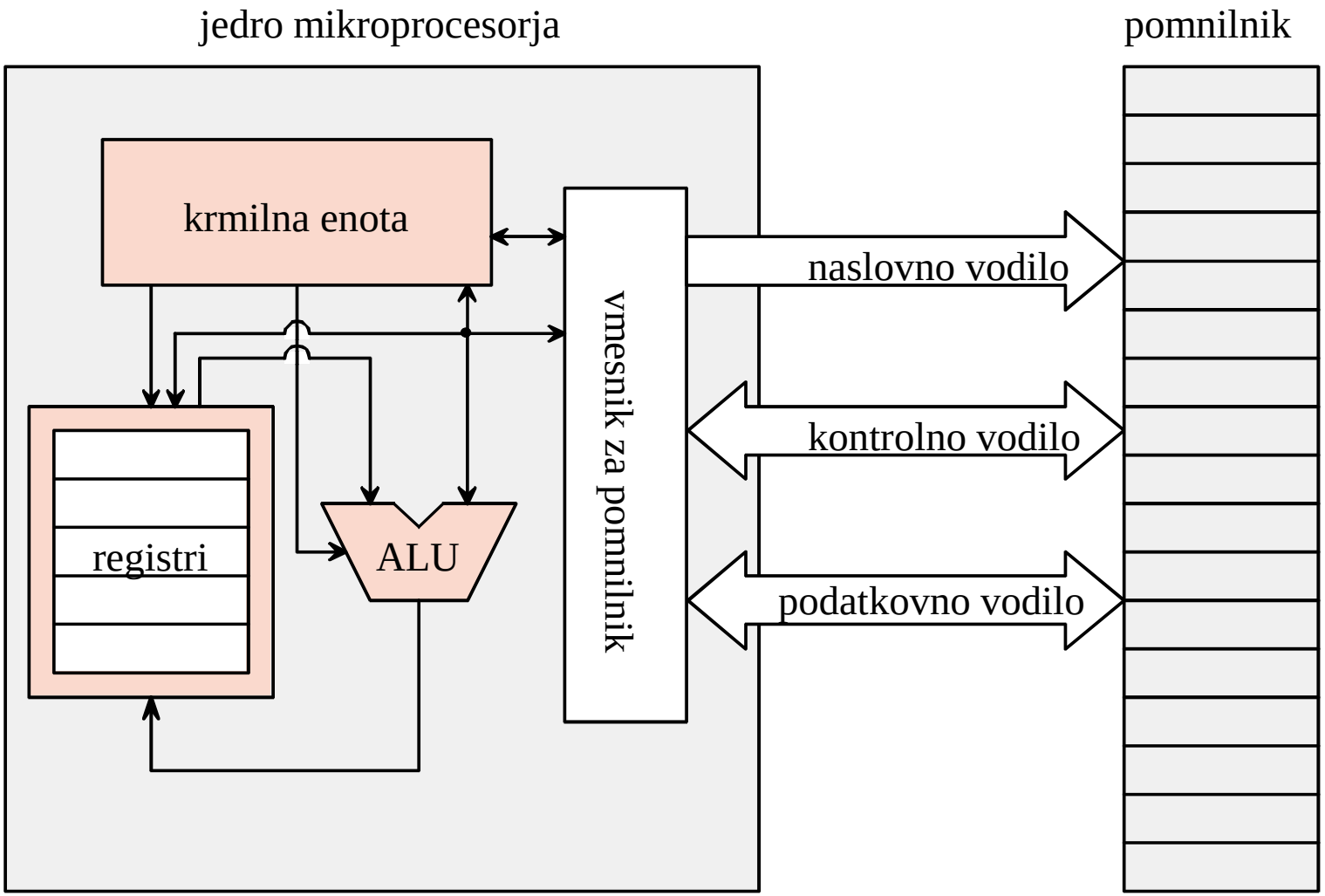
- Organizacija mikroprocesorja je način, s katerim je arhitektura izvedena. Je transparentna za programerja in uporabnike.
  - kontrolni signali,
  - vmesniki,
  - tehnologije

Mikroprocesor (CPU)

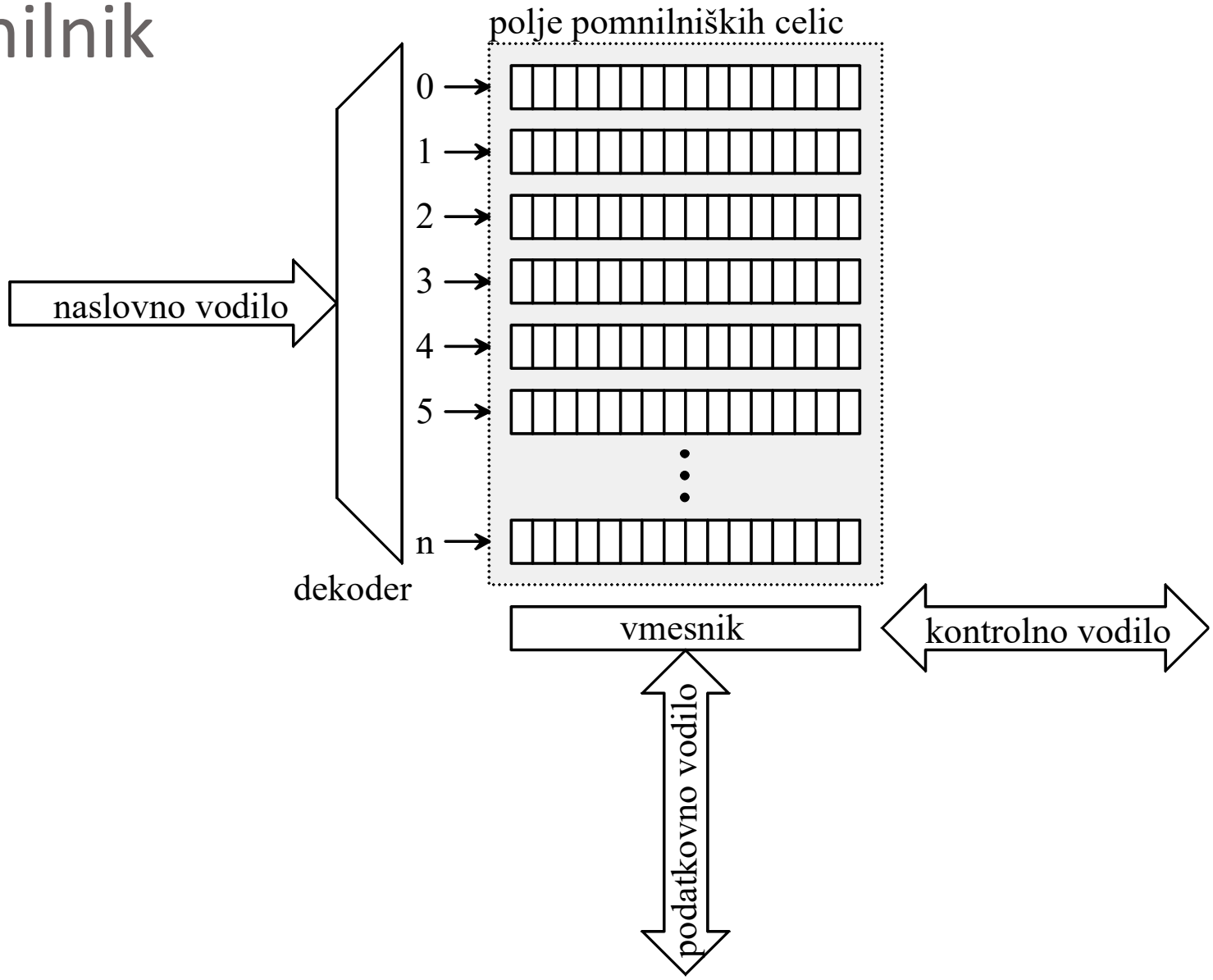


Integrirano vezje, ki sprejema in izvaja kodirane inštrukcije z namenom manipulacije s podatki in kontroliranja vezij, priključenih na vodila.

Von Neumannov model računalnika

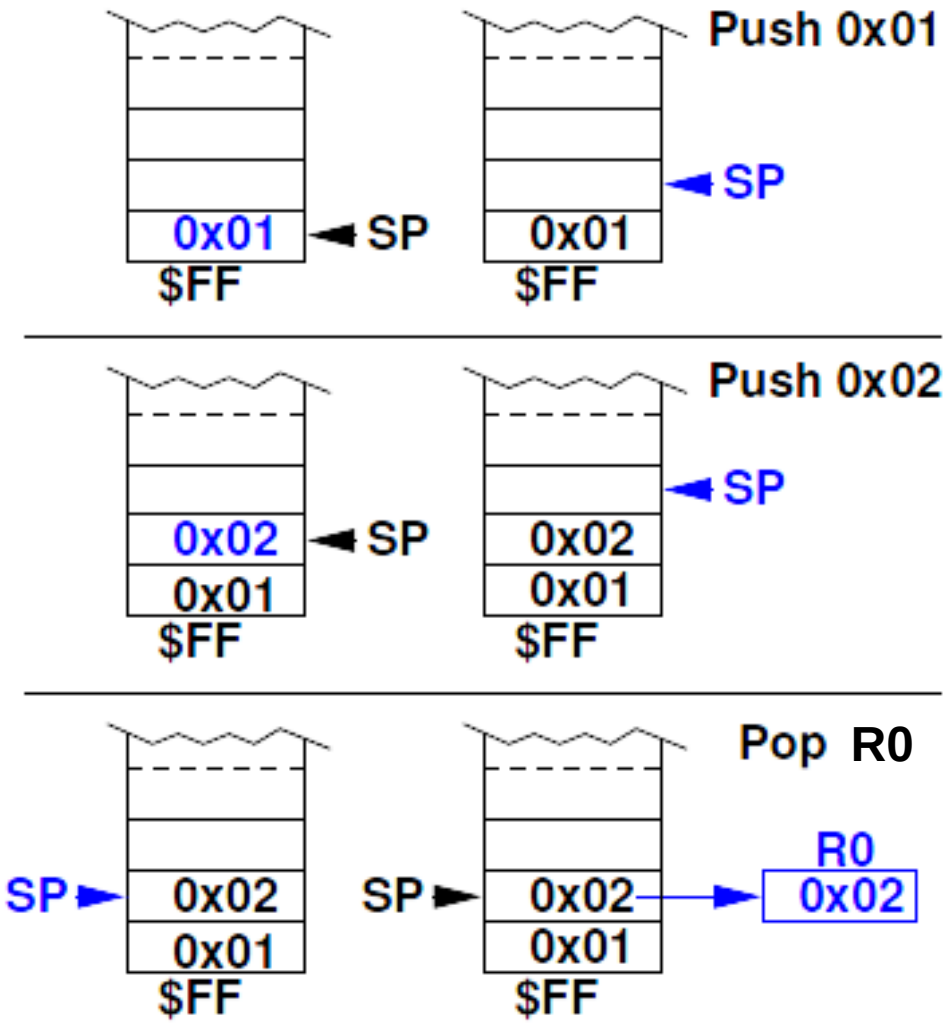


Pomnilnik


polje pomnilniških celic
0
1
2
3
4
5
n
naslovno vodilo
dekoder
vmesnik
kontrolno vodilo
podatkovno vodilo

Sklad (Stack)

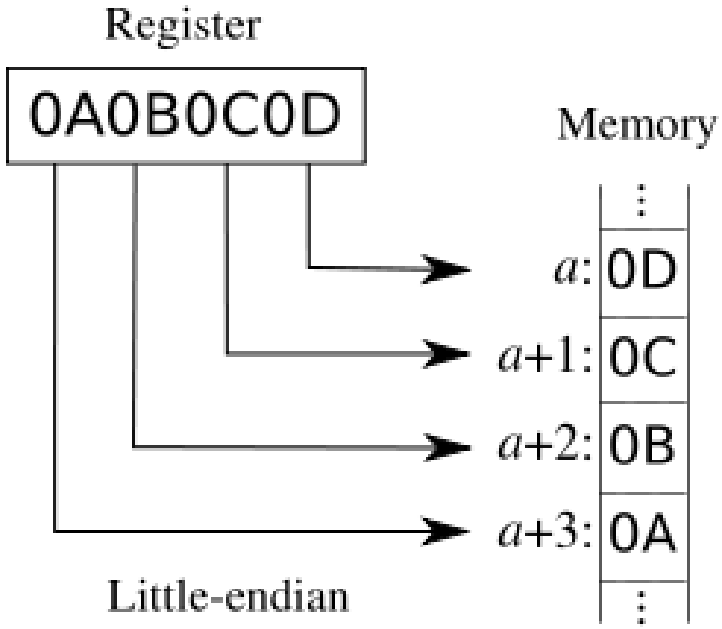
- Del delovnega pomnilnika
  - namenjen začasnemu shranjevanju registrov
- Sistem pomnilnika LIFO
  - Last-In -> first out
  - Ukaza PUSH in POP
- Kazalec na sklad
  - SP (Stack pointer)
  - Vsebuje prvo prazno lokacijo sklada
- Uporaba
  - Klicanje podprogramov
  - Prekinitve

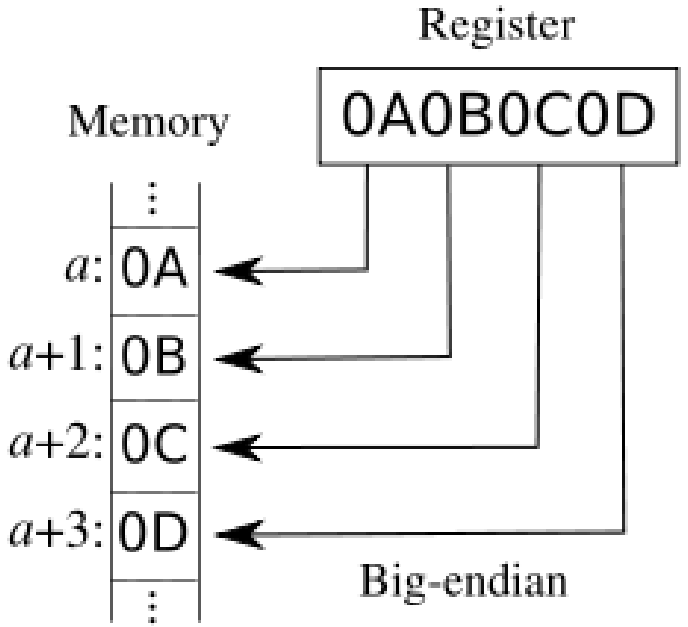


Vrstni red zapisa več-bytnih števil v pomnilniku

Little endian

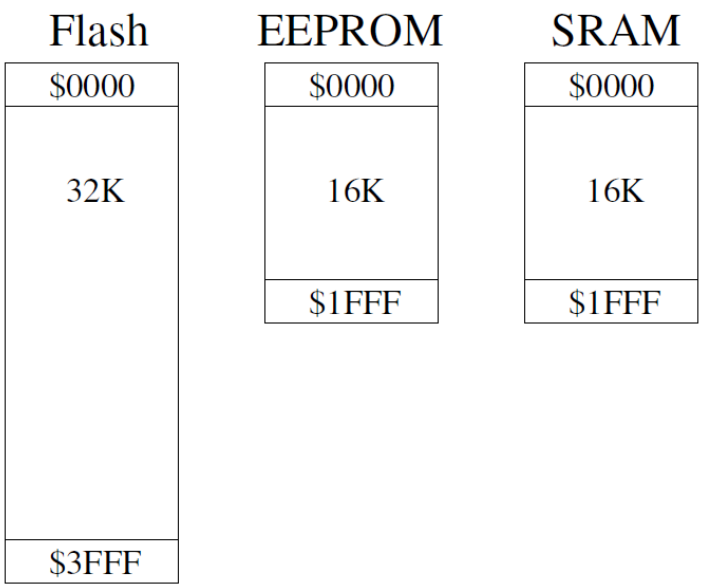
Big endian



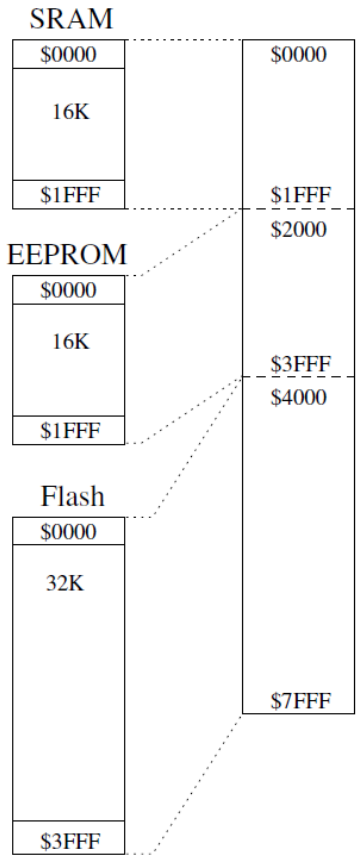


Načini organizacije pomnilnikov

Ločeni naslovni prostori



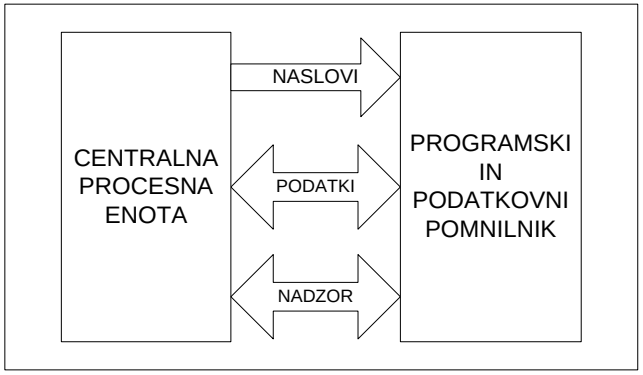
Skupni naslovni prostor



Harvard vs. Princeton

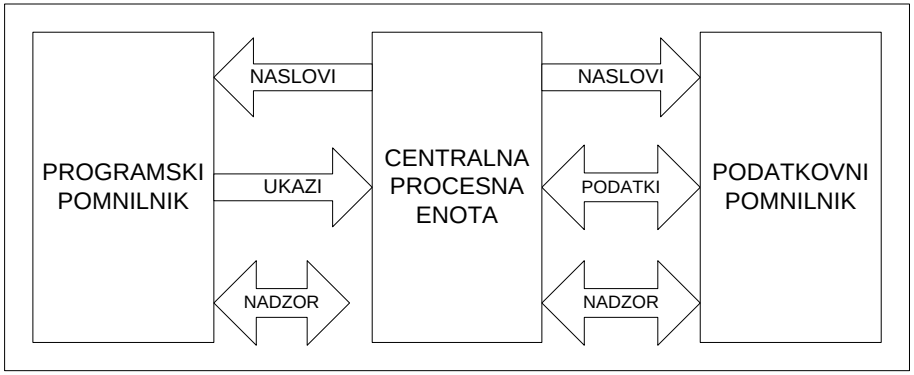
Princetonska arhitektura (Von Neumann)

- Program in podatki v skupnem pomnilniku
- Koda se izvaja zaporedno in zahteva več ciklov
- Program je lahko bolje optimiziran po velikosti
- Intel x86, Pentium, Motorola 68HC11, ARM

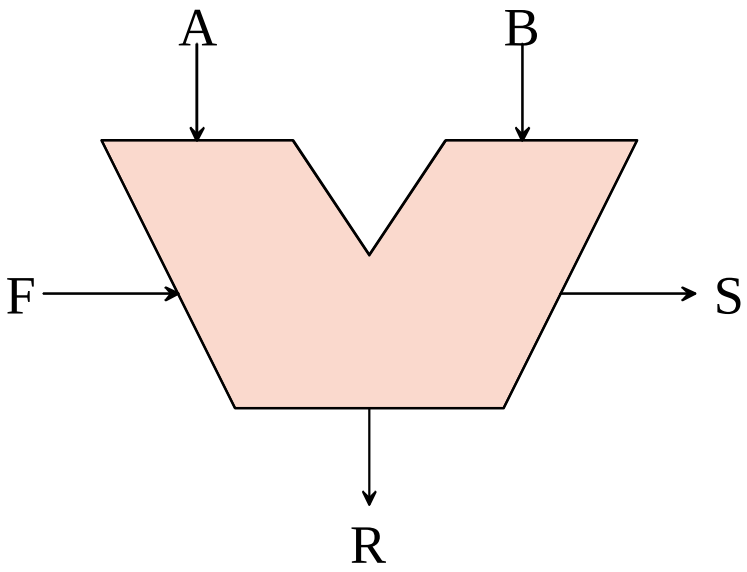


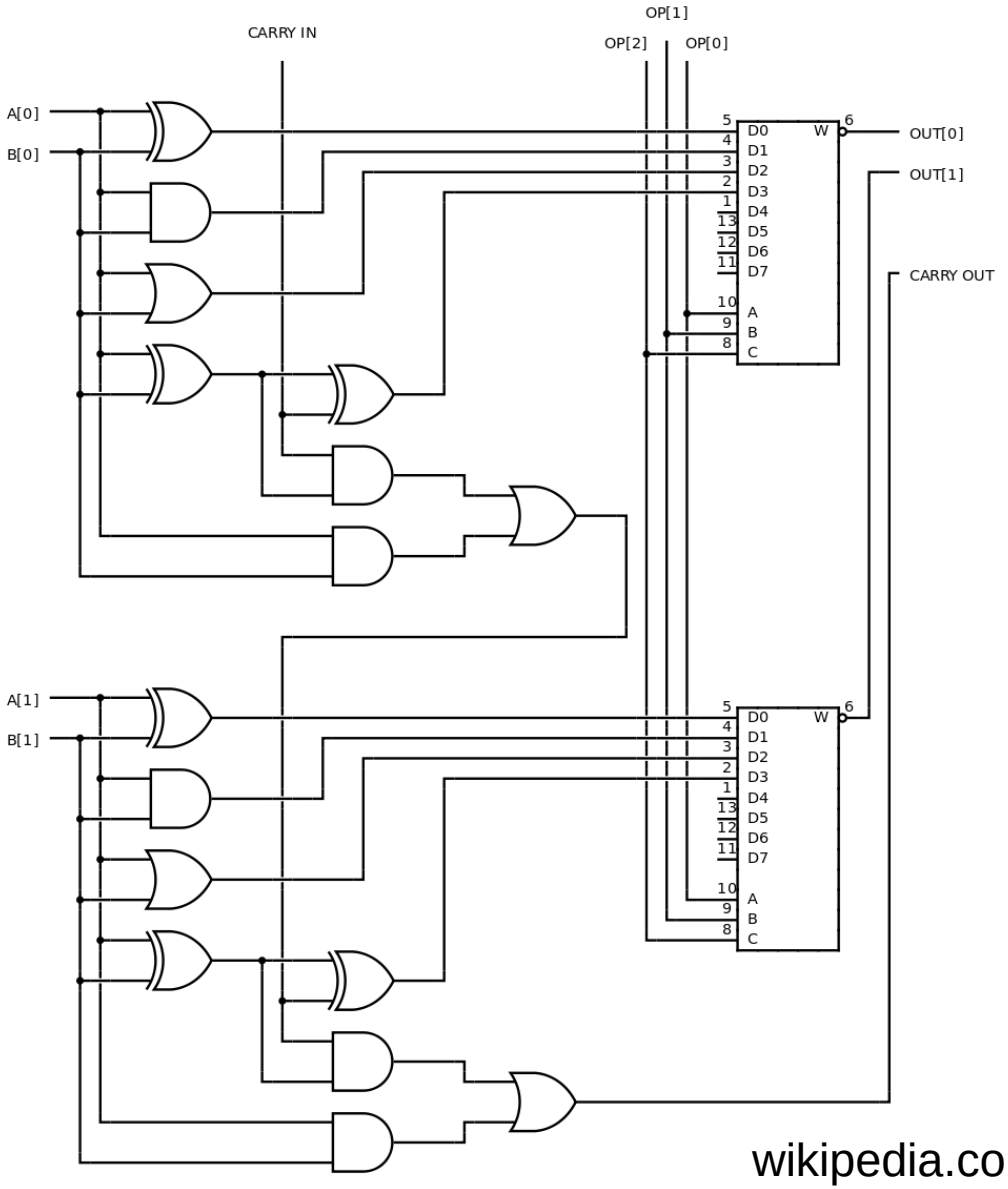
Harvardska arhitektura

- Program in podatki se nahajajo ločeno v vsak svojem pomnilniku
- Koda se lahko izvaja sočasno
- Program ima manj možnosti optimizacije velikosti
- Intel 8051, Atmel AVR, Texas Instruments MSP430, PIC



ALU



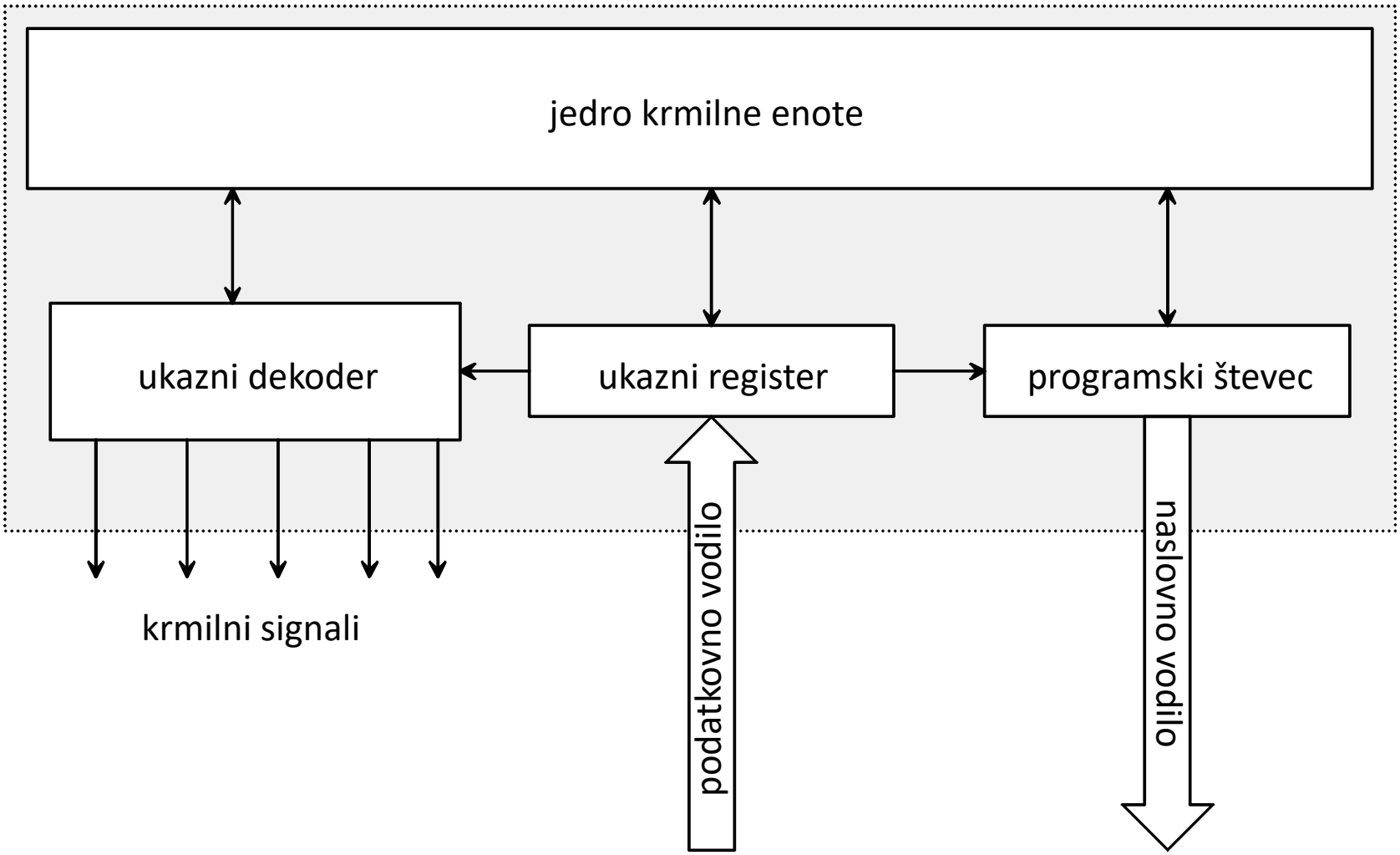


wikipedia.com

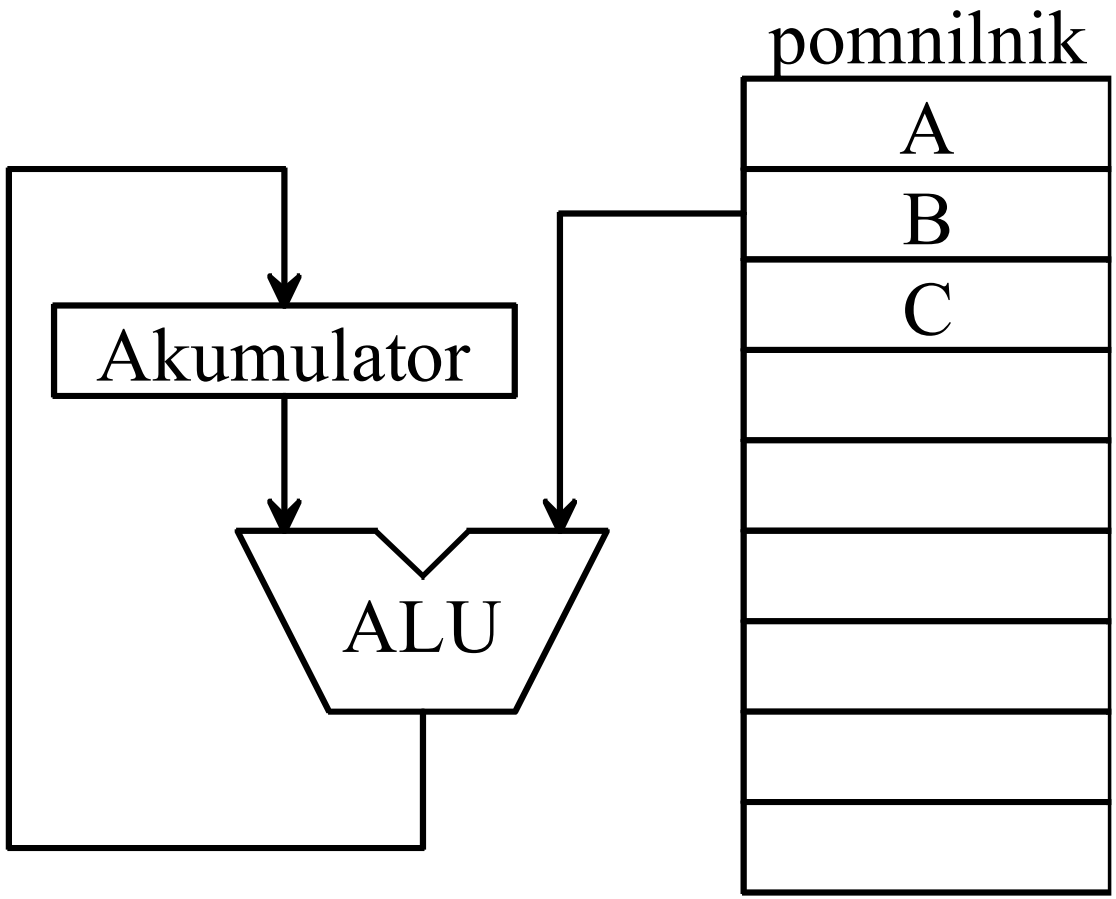
Statusni register

- Vsebuje informacije o zadnji izvedeni aritmetični operaciji
  - Z – rezultat je bil 0
  - N – rezultat je negativen
  - O – prišlo je do preliva (sprememba predznaka)
  - C – prišlo je do prenosa bita
- Uporablja se za preusmerjanje toka programa
  - Posebni ukazi, ki preverjajo posamezen statusni bit in vpišejo zahtevano lokacijo v PC

Krmilna enota



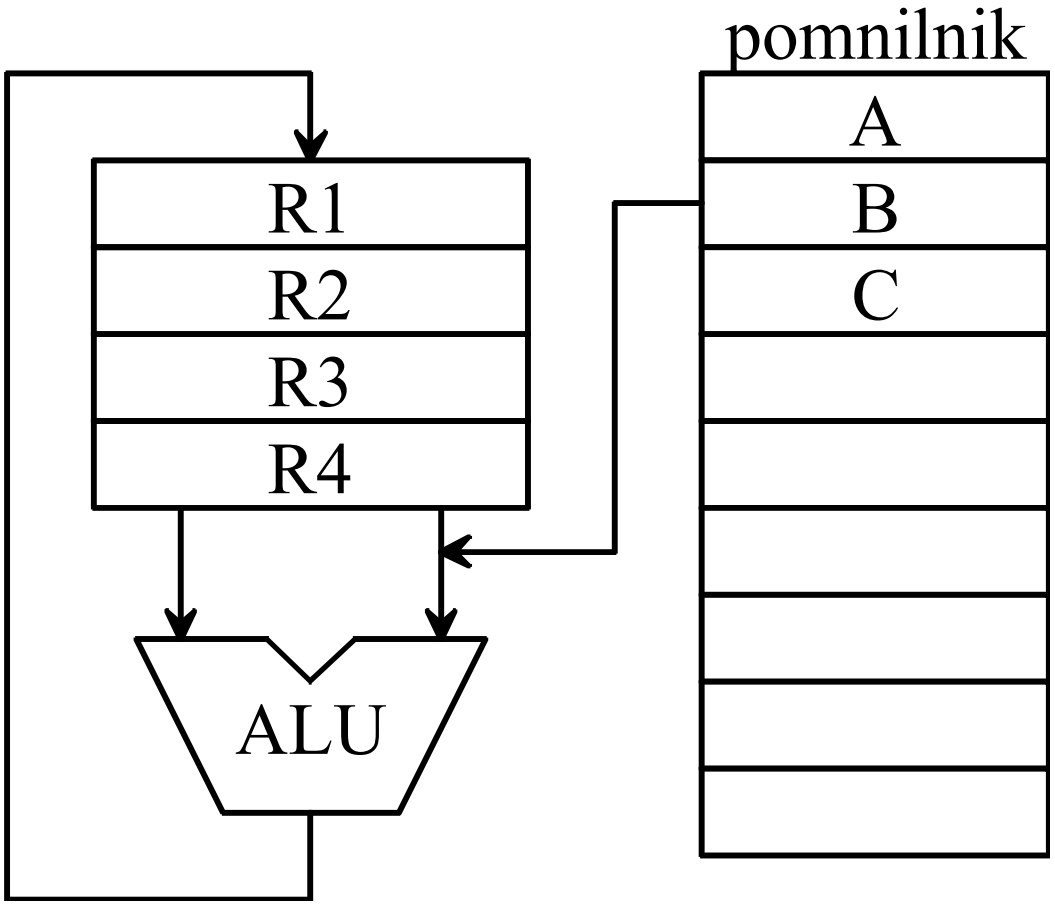
Arhitektura akumulatorja



ukaz	akcija
LOAD A	akum<-A
ADD B	akum<-akum+B
MUL C	akum<-akum*C

akumulator = (A+B)*C

Arhitektura registrov



ukaz	akcija
LOAD R1, A	R1<-A
LOAD R2, B	R2<-B
ADD R1, R2	R1<-R1+R2
LOAD R2, C	R2<-C
MUL R1, R2	R1<-R1*R2
R1=(A+B)*C	

CISC vs. RISC

CISC – Complex Instruction Set Computing

- Dolžine ukazov so različne
- Ukaz se izvede v več ciklih
- Velik nabor ukazov, ki lahko izvajajo kompleksne naloge - mikrokoda
- Kompleksna zgradba procesorja
- Enostavnejše programiranje
- Intel x86, Motorola 68HCxx

RISC – Reduced Instruction Set Computing

- Fiksna dolžina ukaza
- Večina ukazov se izvede v enem urinem ciklu
- Lažja implementacija cevovodov
- Enostavnejša zgradba procesorja, manj tranzistorjev, več prostora za naprednejšo logiko
- Uporaba splošnih registrov za hitri dostop
- Kompleksnejše funkcije se sestavljajo iz osnovnih ukazov
- Kompleksnejše programiranje
- Power PC, Atmel AVR, PIC, ARM

Primer

v register naloži vrednost iz RAM-a na naslovu: 24+x+4*y

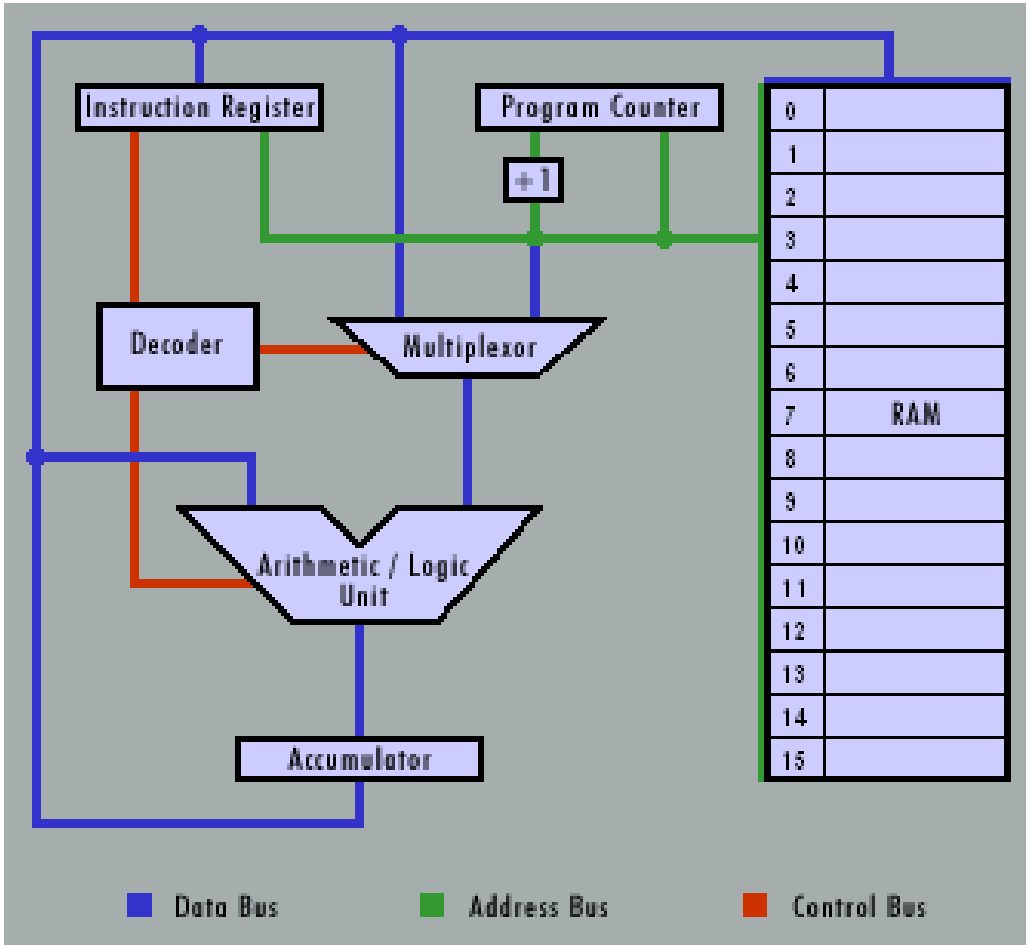
CISC (potrebuje 14 ciklov)

- `MOVE D1, ([24,A0,4*D0])`

RISC (potrebuje 13 ciklov)

- `LD R1, X`
- `LSL R1`
- `LSL R1`
- `MOV X, R0`
- `LD R0, X`
- `ADD R0, R1`
- `LDI R1, $24`
- `ADD R0, R1`
- `MOV X, R0`
- `ST X, R2`

Princip delovanja mikroprocesorja



http://courses.cs.vt.edu/~csonline/MachineArchitecture/Lessons/CPU/index.html

Enostaven strojni jezik

OP koda			Vir	Operand/naslov				
x	x	x	x	x	x	x	x	x

OP koda	Mnemonik	Funkcija	Primer
001	LOAD	Naloži vrednost v akumulator	LOAD 10
010	STORE	Shrani vrednost akumulatorja na dani naslov	STORE 8
011	ADD	Sešteje operand z vrednostjo akumulatorja	ADD #5
100	SUB	Odšteje operand z vrednostjo akumulatorja	SUB #1
101	EQUAL	Če je vrednost operanda enaka akumulatorju, preskoči naslednji ukaz	EQUAL #20
110	JUMP	Skoči na določeno lokacijo izvajanja	JUMP 6
111	HALT	Ustavi program	HALT

Program, ki sešteje dve števili (2+5)

#	Strojni jezik	Zbirniški jezik	Opis
0	001 1 000010	LOAD #2	Naloži konstanto 2 v akumulator
1	010 0 001101	STORE 13	Shrani akumulator v spremenljivko na lokaciji13
2	001 1 000101	LOAD #5	Naloži konstanto 5 v akumulator
3	010 0 001110	STORE 14	Shrani akumulator v spremenljivko na lokaciji 14
4	001 0 001101	LOAD 13	Naloži spremenljivko na lokaciji 13 v akumulator
5	011 0 001110	ADD 14	Sešteje vrednost akumulatorja z lokacijo 14
6	010 0 001111	STORE 15	Shrani akumulator v spremenljivko na lokaciji 15
7	111 0 000000	HALT	Ustavi izvajanje programa

Animacije

- Vse animacije si lahko ogledate na naslednji povezavi:
  - https://courses.cs.vt.edu/~csonline/MachineArchitecture/Lessons/CPU/index.html
- Stran lahko odprete le s Flash browserjem, ki si ga lahko naložite s tukaj: https://github.com/radubirsan/FlashBrowser/releases/tag/v0.8