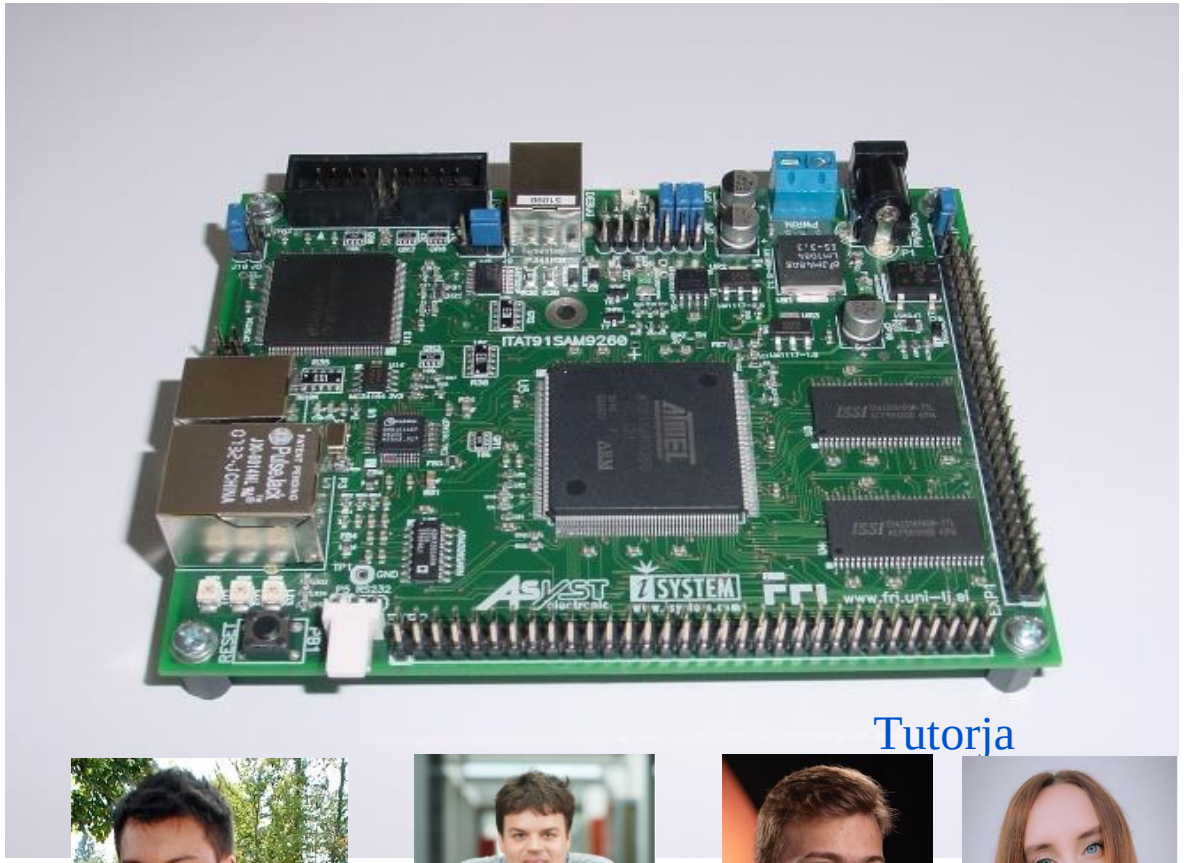
Računalniška arhitektura RA



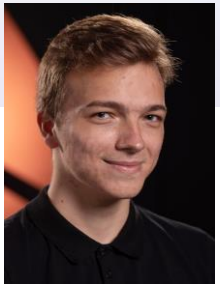
Ekipa RA

Mira Trebar
mira.trebar@fri...

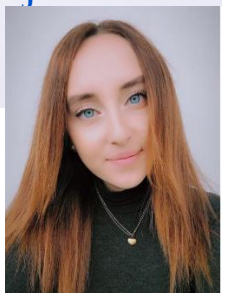
Žiga Pušnik
ziga.pusnik@fri....

Rok Češnovar
Rok.cesnovar@fri...

Tutorja



Miha Krajnc
mk7793@student.uni-lj.si



Anamari Orehar
ao6477@student.uni-lj.si

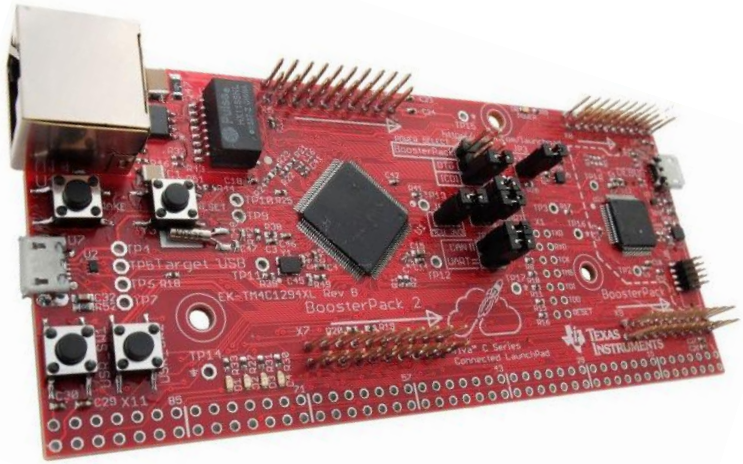
Robert Rozman
rozman@fri.uni-lj.si

Računalniška arhitektura RA

LAB 1.1 Splošne informacije

Laboratorijske vaje RA

- Spoznati osnove računalniške arhitekture s praktičnega vidika
- Razumeti delovanje računalnika (ARM) s programiranjem v zbirnem jeziku
- Podrobnejši vpogled:
  - v delovanje računalnika
  - v izvajanje programov na računalniku
- Vsebinske nadgradnje -> predmet Organizacija računalnikov in ostali



Vsebina vaj

- Potrebne osnove s predavanj (npr. pomnilniški naslov, vsebina, …)
- Jedro: Programiranje v zbirnem jeziku ARM
- Oblika: Sprotne vaje + domača naloga
- Tri preverjanja* (november, december, januar)
- Priprava na izpit (avditorne naloge)

- Predmetni seminar po dogovoru z asistentom

**V primeru izrednih razmer se lahko spremeni*

Ocenjevanje*

Vaje prispevajo 50% h končni oceni in morajo biti opravljene naslednje obveznosti:

- Uspešno opraviti sprotne naloge in biti prisoten na laboratorijskih vajah
- Uspešno oddati in zagovarjati domačo nalogo,
- Tri preverjanja (80 + 100 + 120 točk)
  - skupaj potrebno zbrati vsaj 150 točk (50%)
  - ni omejitev na posameznih preverjanjih
  - *v primeru „Covid zapore“ se 1. in 2. preverjanje spremenita v domači nalogi in se 3. preverjanje opravi v okviru pisnega izpita in/ali ustnega izpita
- Ocena vaj velja le v tekočem študijskem letu. Kdor v istem letu ne opravi predmeta v celoti, mora prihodnje leto ponovno opraviti vaje.

**V primeru izrednih razmer se lahko način ocenjevanja spremeni*

Razvojno okolje WinIDEA

simpr - winIDEA - [C:\winIDEA\Projekti\Zgled_2020\user.s]

File View Project Simulator Debug Test Plugins Tools Window Help

Project Workspace

Filter

sample.elf

Symbols Project

user.s crt0.s sample.lcf

```
stev2:    .word 0x10
rez:      .space 4

          .align
          .global __start
__start:

          ldr r1, stev1
          ldr r2, stev2
          add r3, r2, r1
          str r3, rez

__end:    b __end
```

Memory 0x0000000

Area: Virtual Address: 0x0000000 Symbol

00000000	09	00	00	EA	08	00	00	EA
00000008	07	00	00	EA	06	00	00	EA
00000010	05	00	00	EA	04	00	00	EA
00000018	03	00	00	EA	02	00	00	EA
00000020	40	00	00	00	10	00	00	00
00000028	00	00	00	00	14	10	1F	E5
00000030	14	20	1F	E5	01	30	82	E0
00000038	18	30	0F	E5	FE	FF	FF	EA
00000040	00	00	00	00	00	00	00	00
00000048	00	00	00	00	00	00	00	00
00000050	00	00	00	00	00	00	00	00
00000058	00	00	00	00	00	00	00	00
00000060	00	00	00	00	00	00	00	00

Disassembly

__start

Address	Data	Disassembly
		__start
		ldr r1, stev1
000(	14101	ldr r1,[pc,-0014]
		ldr r2, stev2
000(	14201	ldr r2,[pc,-0014]
		add r3, r2, r1
000(	01308	add r3,r2,r1
		str r3, rez
000(	18300	str r3,[pc,-0018]
		__end: b __end

Registers

R0	00000000
R1	00000000
R2	00000000
R3	00000000
R4	00000000
R5	00000000
R6	00000000
R7	00000000
R8	00000000
R9	00000000

Output

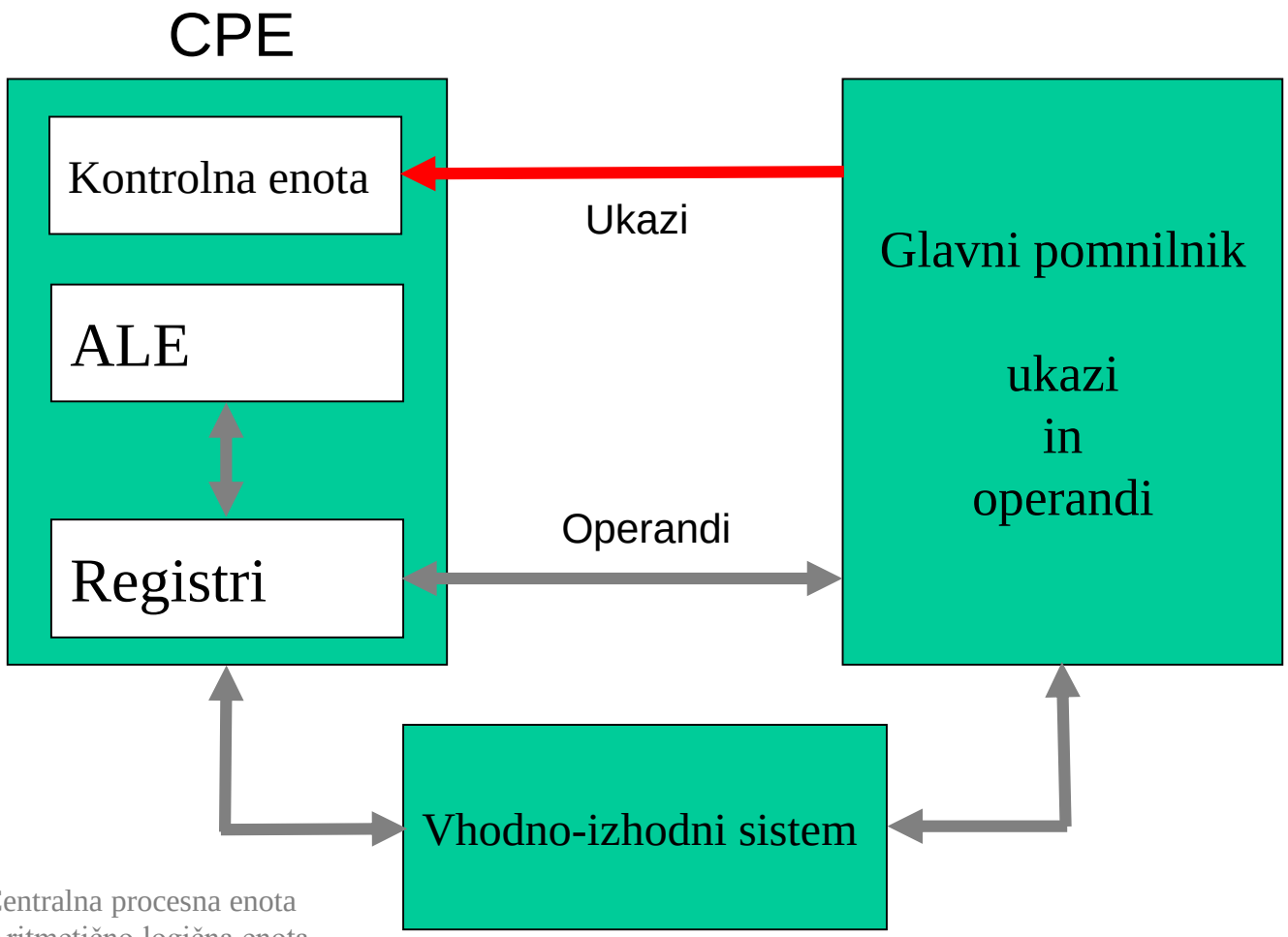
```
Compiling ...
crt0.s
user.s
Linking
"sample.elf (Dir:C:\winIDEA\Projekti\Zgled_2020\Debug\)"  ... was successfully generated.
0 Error(s)  0 Warning(s)
```

Build Find In Files Tools Script

Računalniška arhitektura RA

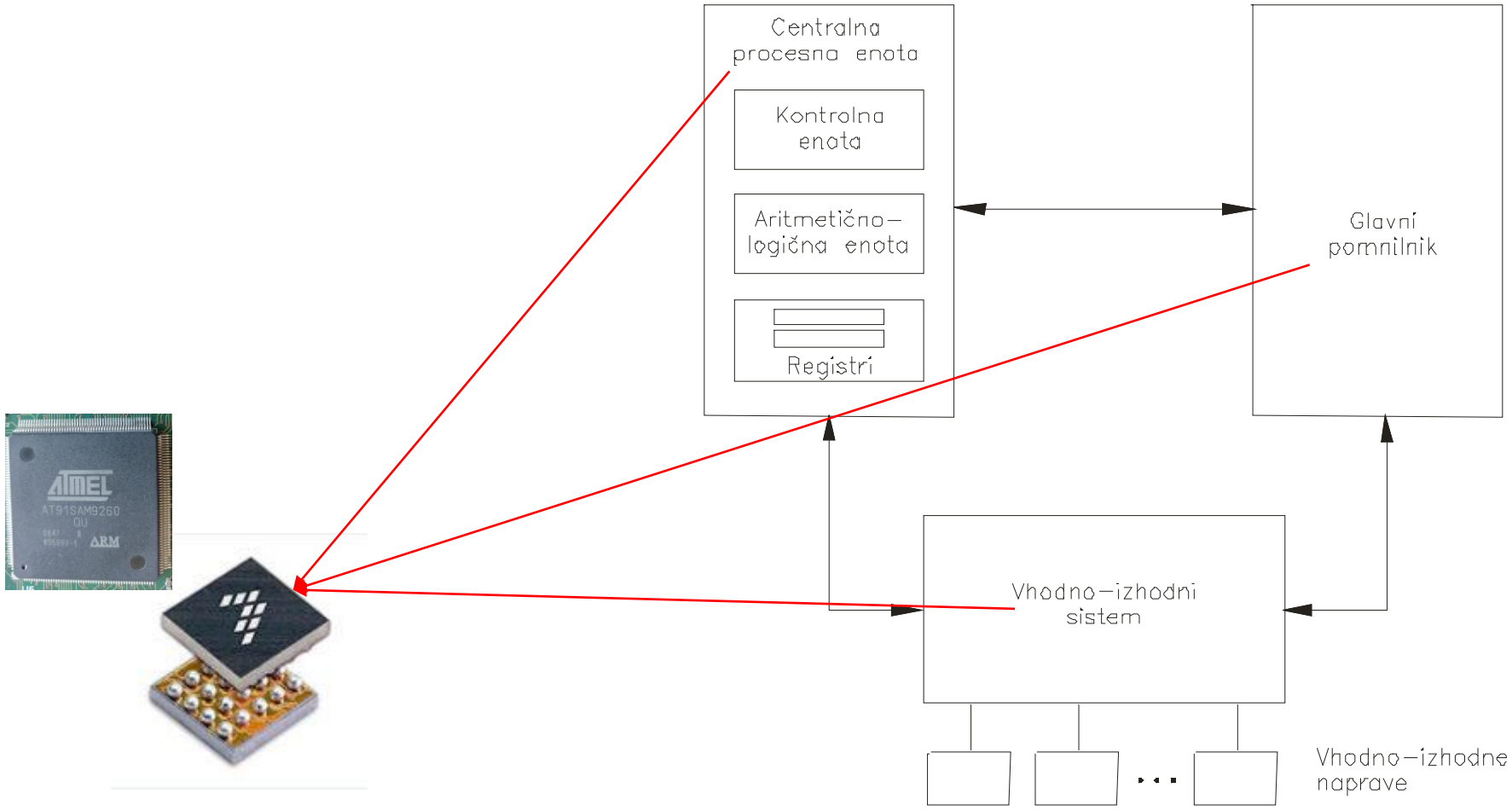
LAB 1.2 Von Neumannov model (VN)

Von Neumannov računalniški model



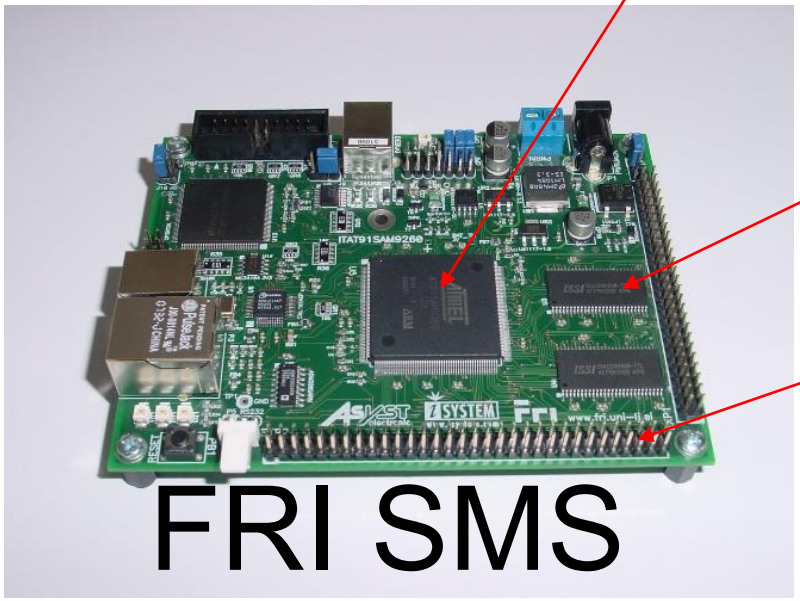
CPE – Centralna procesna enota
ALE – Aritmetično logična enota

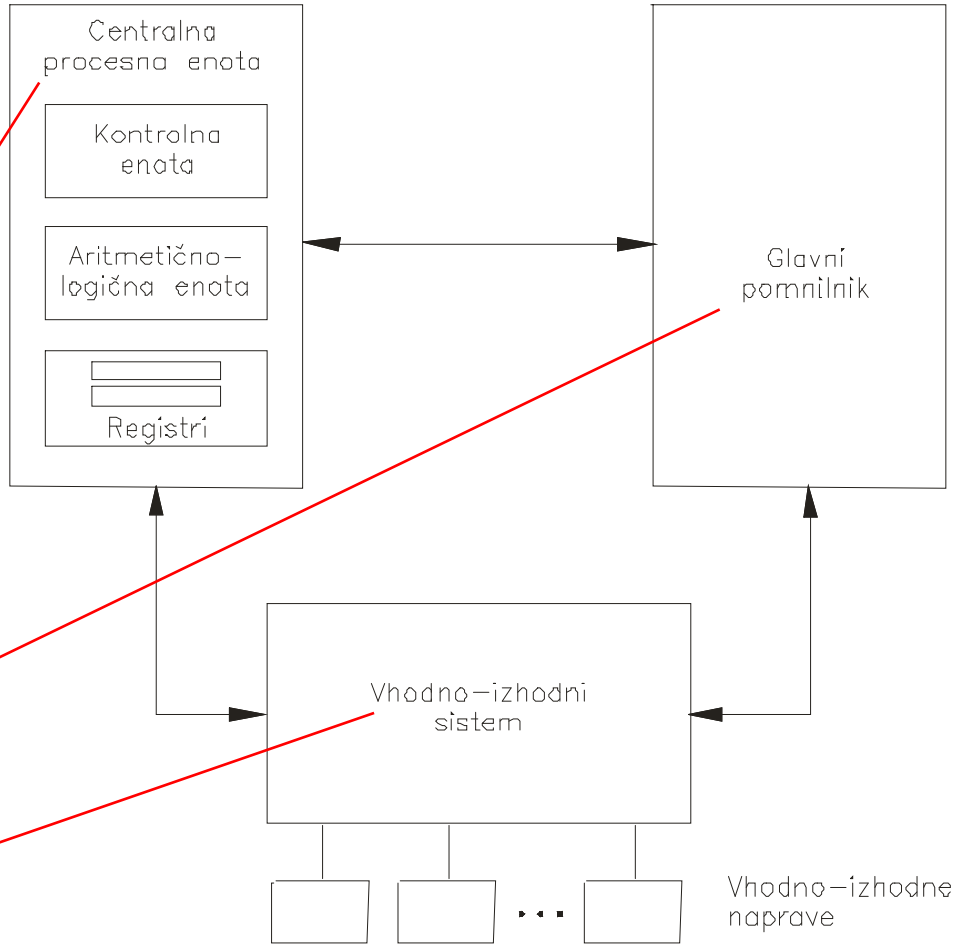
Osnovni model računalnika



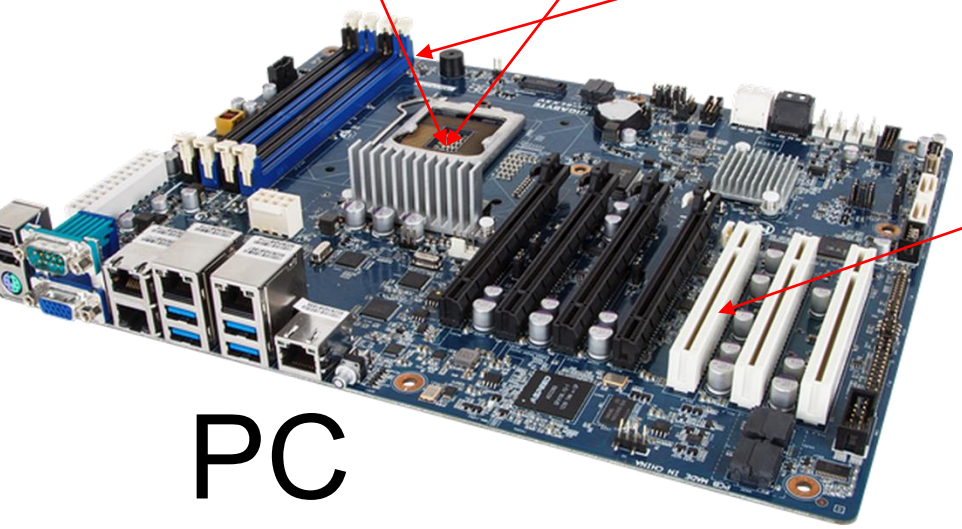
Mikrokrmilniki

Osnovni model računalnika

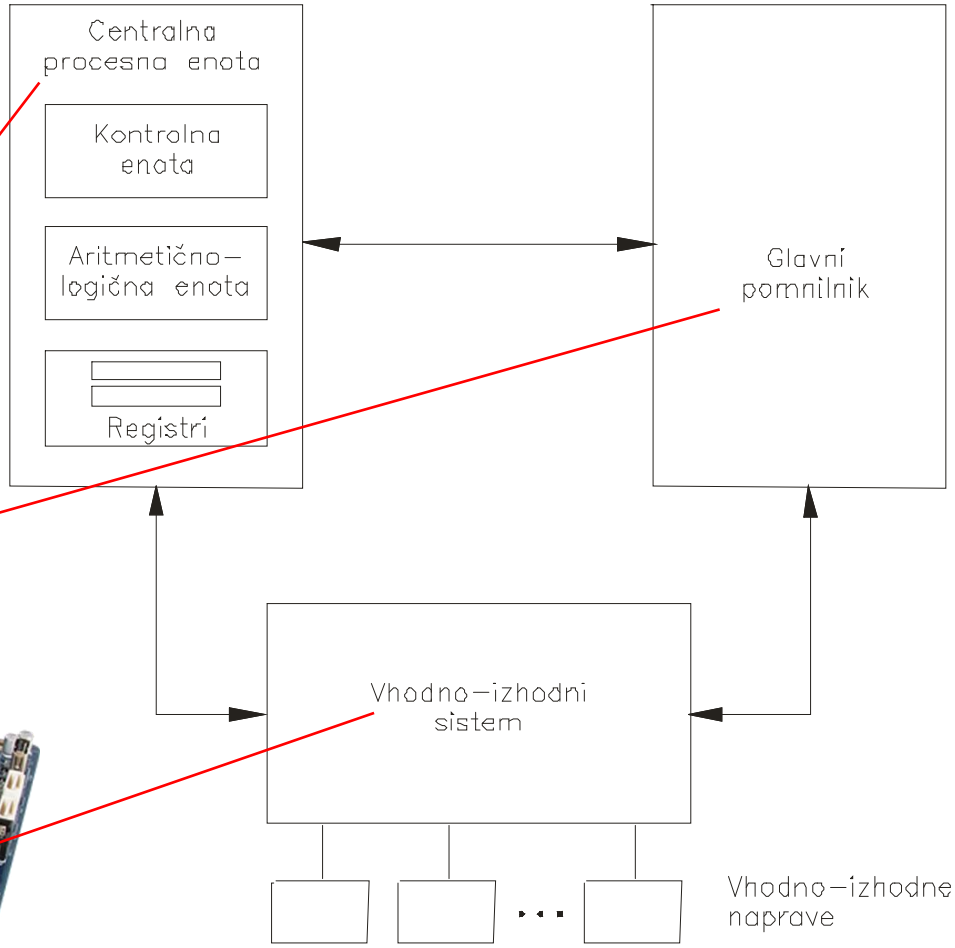




Osnovni model računalnika



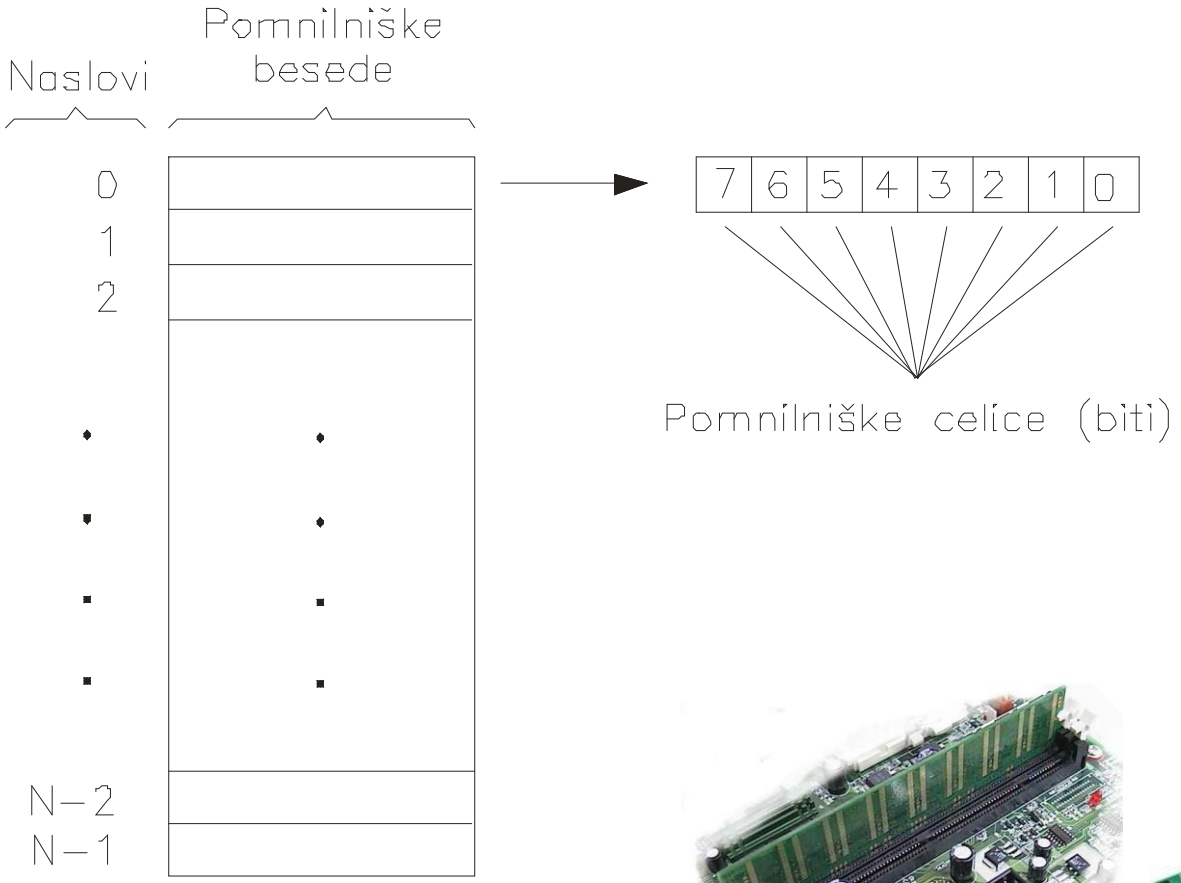
PC

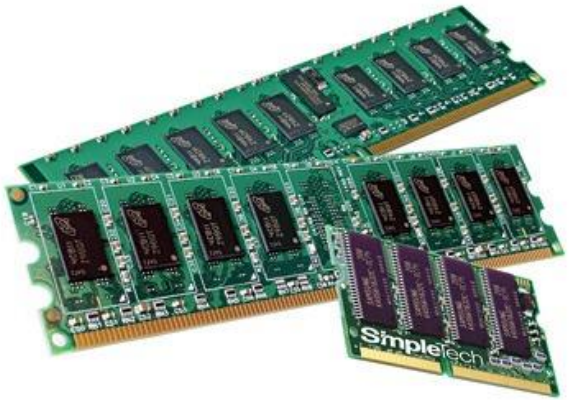


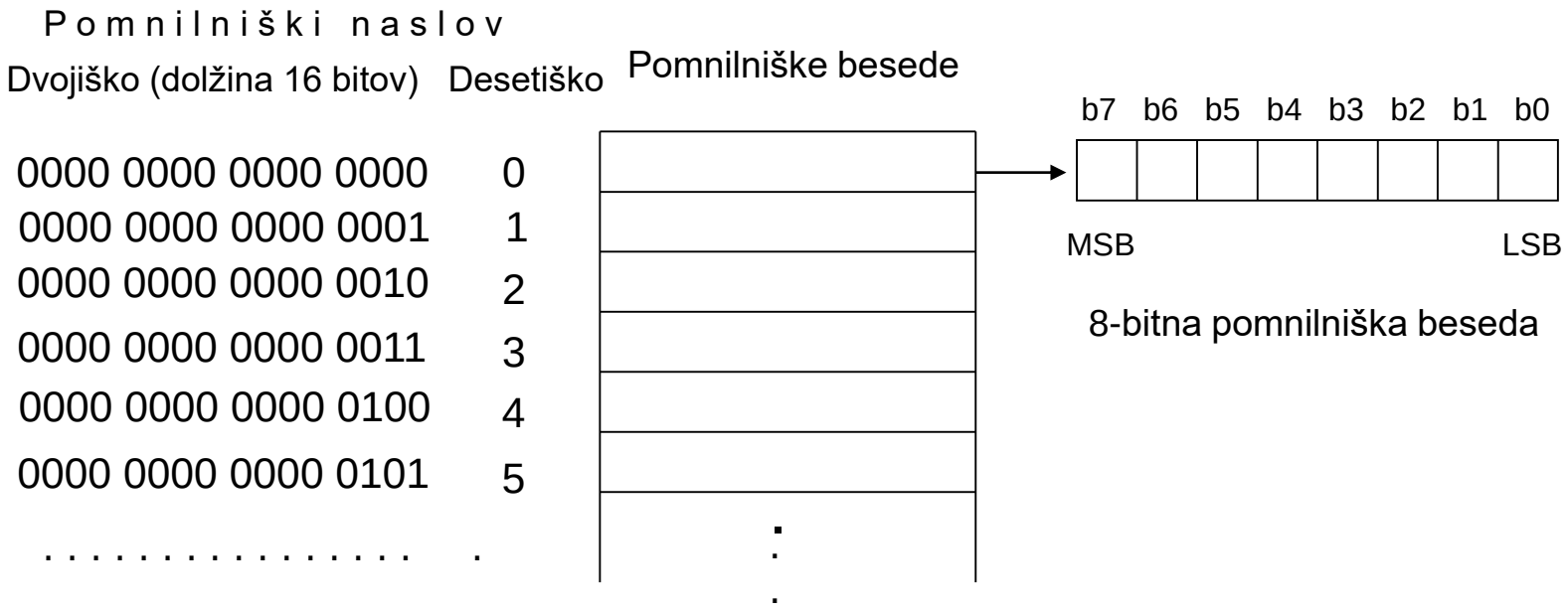
Računalniška arhitektura RA

LAB 1.3 Pomnilnik

Kaj je pomnilnik ?






P o m n i l n i š k i n a s l o v
Dvojiško (dolžina 16 bitov)
Desetiško
Pomnilniške besede
0000 0000 0000 0000
0
0000 0000 0000 0001
1
0000 0000 0000 0010
2
0000 0000 0000 0011
3
0000 0000 0000 0100
4
0000 0000 0000 0101
5
b7 b6 b5 b4 b3 b2 b1 b0
MSB
LSB
8-bitna pomnilniška beseda

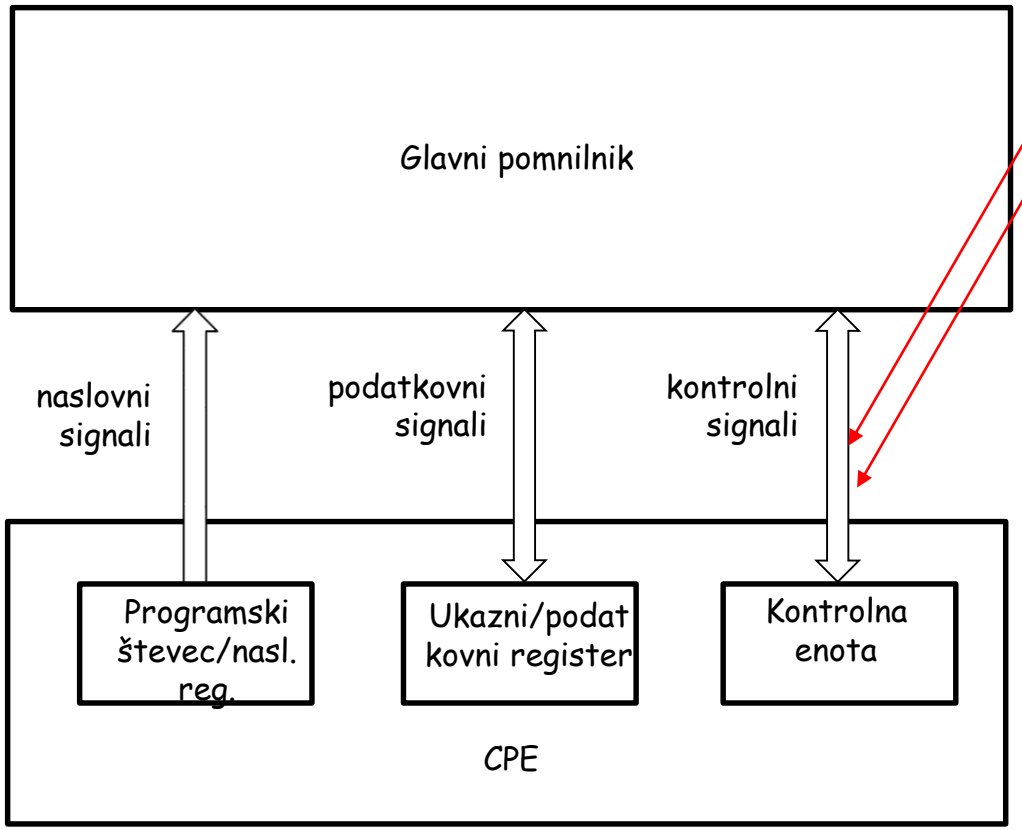
P o m n i l n i š k i n a s l o v

Dvojiško (dolžina 16 bitov)	Šestnajst.	Desetiško	Pomnilniške besede
0000 0000 0000 0000	0000	0	
0000 0000 0000 0001	0001	1	
0000 0000 0000 0010	0002	2	
0000 0000 0000 0011	0003	3	
0000 0000 0000 0100	0004	4	
0000 0000 0000 0101	0005	5	
. . . . .	.		
. . . . .	.		
1111 1111 1111 1011	FFFB	65531	
1111 1111 1111 1100	FFFC	65532	
1111 1111 1111 1101	FFFD	65533	
1111 1111 1111 1110	FFFE	65534	
1111 1111 1111 1111	FFFF	65535	

Povezava CPE <-> glavni pomnilnik

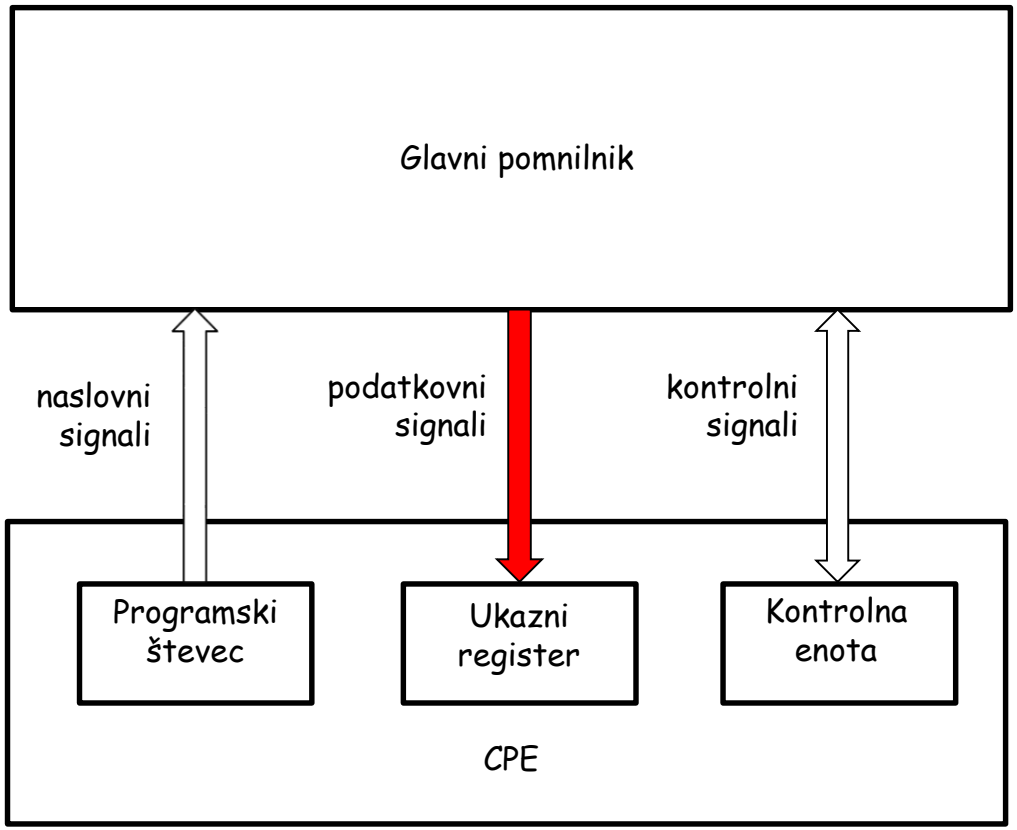
Vodilo = skupina povezav (naslovno, podatkovno, kontrolno, …)

Linija = povezava
Signal = vsebina, ki se prenaša po povezavi (1bit)



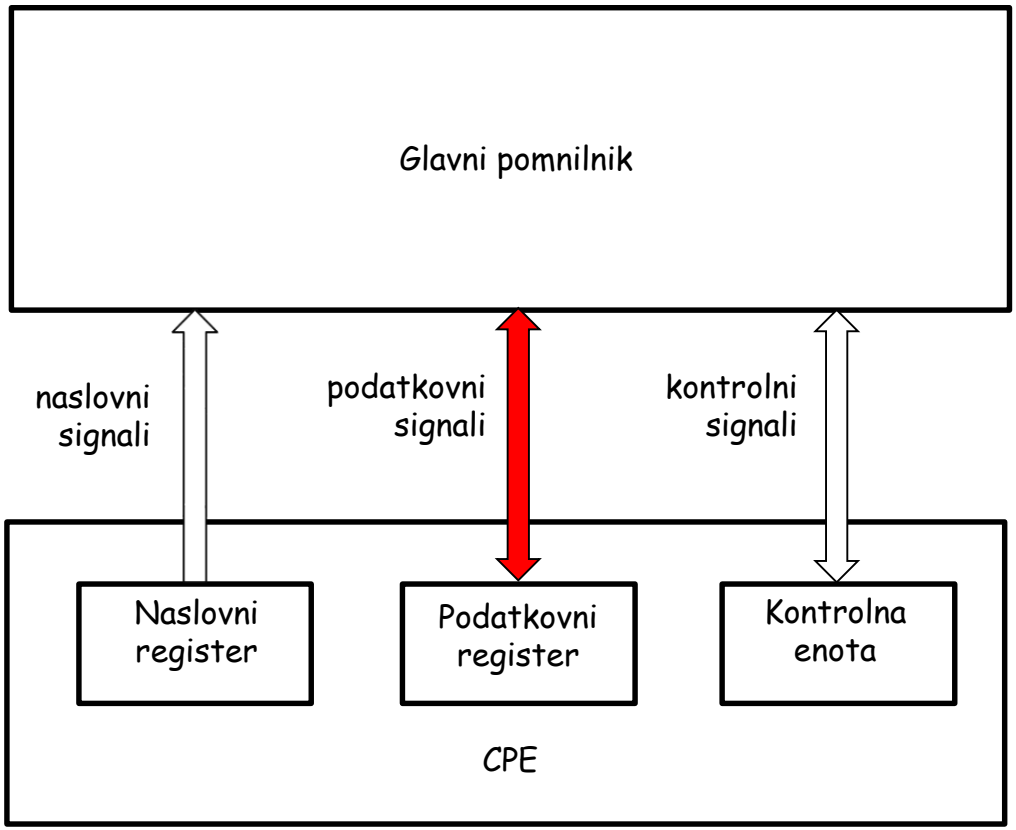
Kako CPE dostopa do glavnega pomnilnika?

Primer za ukaze:

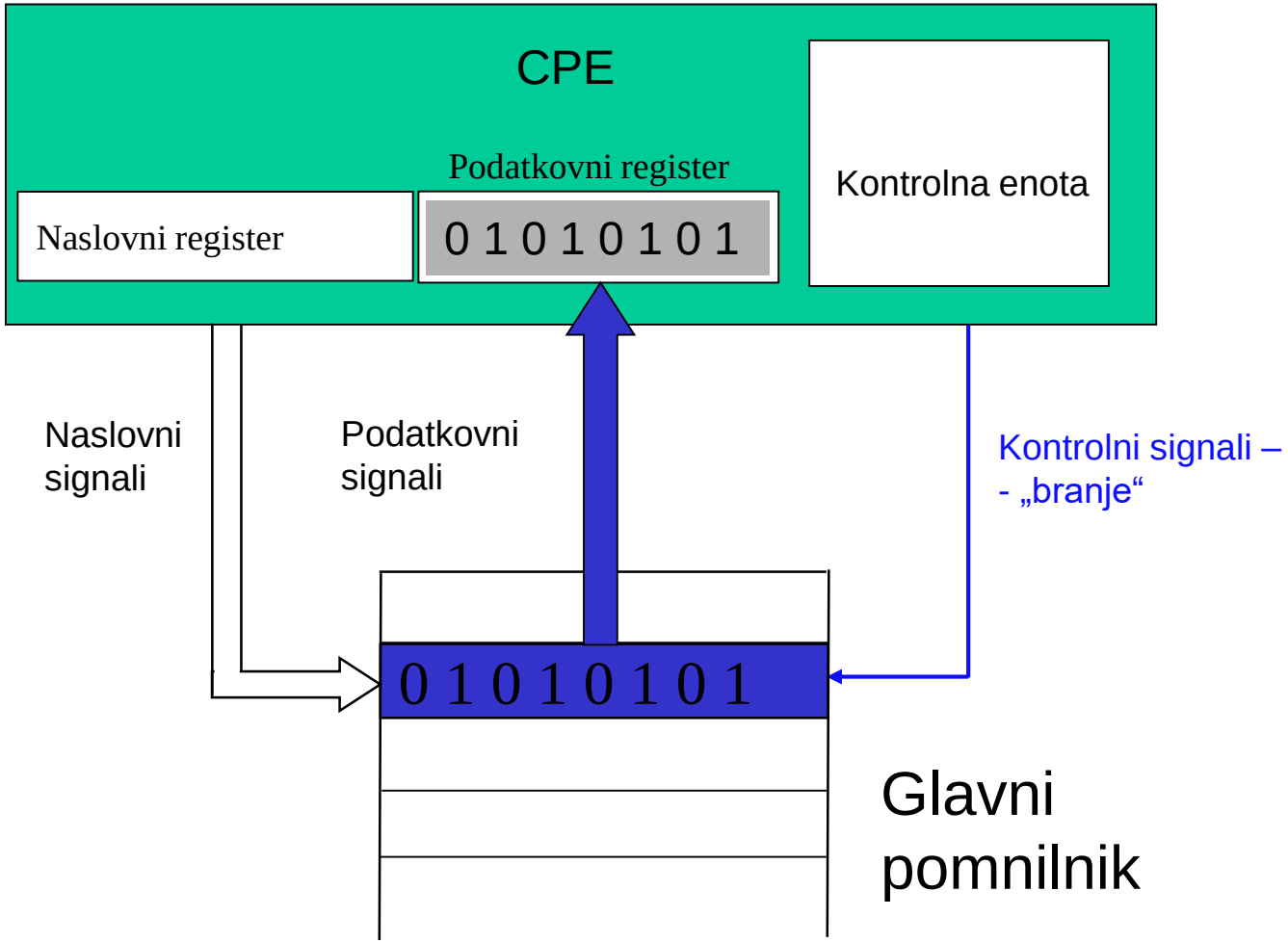


Kako CPE dostopa do glavnega pomnilnika?

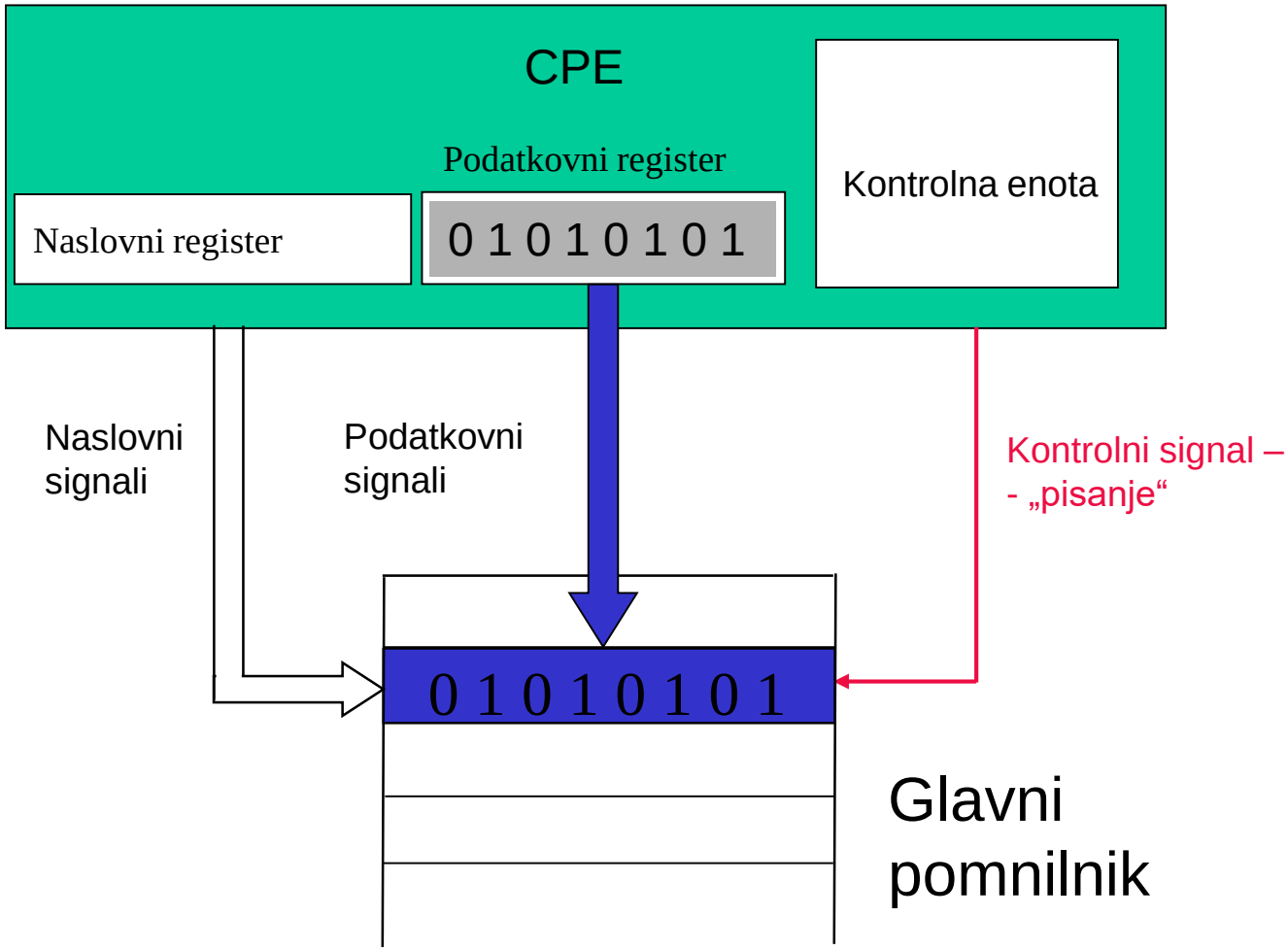
Primera za operande:



Povezava med CPE in glavnim pomnilnikom – bralni dostop



Povezava med CPE in glavnim pomnilnikom – pisalni dostop



Računalniška arhitektura RA

LAB 1.4 Številski sistemi (BIN,HEX) na hitro

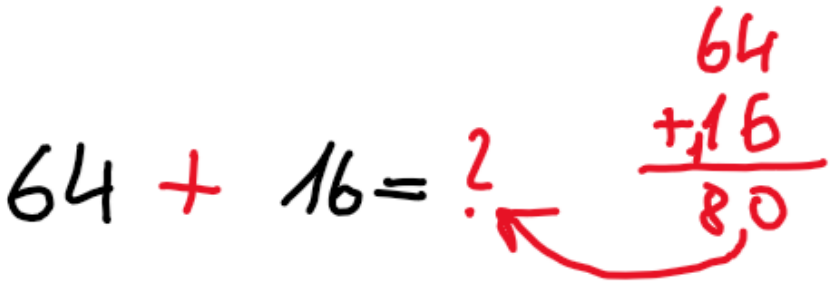
Računalniška arhitektura RA

LAB 1.5 Pravilo tankega/debelega konca na hitro

Računalniška arhitektura RA

LAB 1.6 Seštevanje – človek, python, zbirnik

Človek (zgled: 64 + 16 = 80)



Python (zgled: REZ = STEV1 + STEV2)

Seštevanje spremenljivk v Pythonu.

http://goo.gl/YXQ5qN

Python 2.7

```
1  STEV1=0x40
2  STEV2=0x10
3  REZ = STEV1 + STEV2
4  print (" STEV1 = " + hex(STEV1) + "\n+STEV2 = " + hex(STE
```

Frames Objects

Global frame	
STEV1	64
STEV2	16
REZ	80

Print output (drag lower right corner to resize)

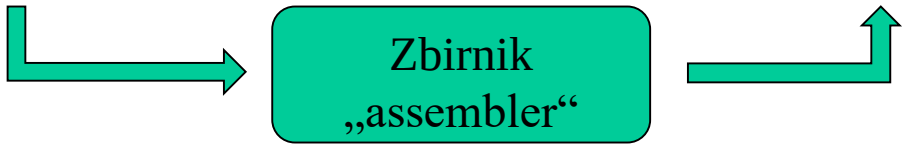
```
 STEV1 = 0x40
+STEV2 = 0x10
-------------
   REZ = 0x50
```

Zbirni jezik (zgled: rez=stev1+stev2)

Seštevanje spremenljivk v zbirniku ARM. Prenesite pripravljen projekt na e-učilnici.

Vrednosti spremenljivk so shranjene v pomnilniku.
Operacije realiziramo s programom z naslednjimi ukazi:

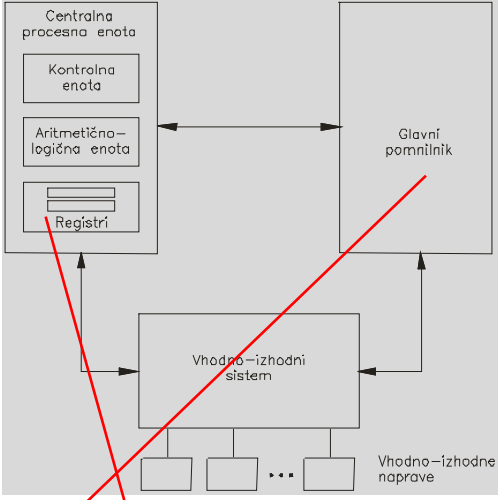
Zbirni jezik	Opis ukaza	Strojni jezik
ldr r1, stev1	R1 ← M[0x20]	0xE51F1014
ldr r2, stev2	R2 ← M[0x24]	0xE51F2014
add r3, r2, r1	R3 ← R1 + R2	0xE0823001
str r3, rez	M[0x28] ← R3	0xE50F3018



Ukaze izvajajte po korakih in opazujte vrednosti registrov in vrednosti spremenljivk v pomnilniku.

Praktično delo: Zgled

Razvojno okolje WinIDEA



simpr - winIDEA - [C:\winIDEA\Projekti\Zgled_2020\user.s]

File View Project Simulator Debug Test Plugins Tools Window Help

Project Workspace

Filter

sample.elf

user.s | crt0.s | sample.lcf

```
stev2:      .word 0x10
rez:        .space 4

            .align
            .global __start
__start:

            ldr r1, stev1
            ldr r2, stev2
            add r3, r2, r1
            str r3, rez
```

Memory 0x0000000

Area: Virtual Address: 0x0000000 Symbol

```
00000000  09 00 00 EA 08 00 00 EA
00000008  07 00 00 EA 06 00 00 EA
00000010  05 00 00 EA 04 00 00 EA
00000018  03 00 00 EA 02 00 00 EA
00000020  40 00 00 00 10 00 00 00
00000028  00 00 00 00 14 10 1F E5
00000030  14 20 1F E5 01 30 82 E0
00000038  18 30 0F E5 FE FF FF EA
00000040  00 00 00 00 00 00 00 00
00000048  00 00 00 00 00 00 00 00
00000050  00 00 00 00 00 00 00 00
00000058  00 00 00 00 00 00 00 00
00000060  00 00 00 00 00 00 00 00
```

Registers

```
R0 00000000
R1 00000000
R2 00000000
R3 00000000
R4 00000000
R5 00000000
R6 00000000
R7 00000000
R8 00000000
R9 00000000
```

Razvojno okolje WinIDEA (dokumentacija, začetni projekti)

- Instalacija orodja WinIdea - Windows
- Video: Odpiranje začetnega projekta za simulator
- Začetni projekt za winIDEA (simulator)
- Začetni projekt za winIDEA 2016 (ploščica)

Skrito za udeležence

- Instalacija orodja WinIdea - Linux
- Seznam ukazov zbirnika ARM
- ARM sistem FRI-SMS (ploščica)

Skrito za udeležence

- WinIdea: Software User Guide
- Povezava na online Help za WinIdea

Download

Računalniška arhitektura RA

LAB 1.7 Predloge za zapiske

Python (zgled: REZ = STEV1 + STEV2)

Frames Objects

Global frame

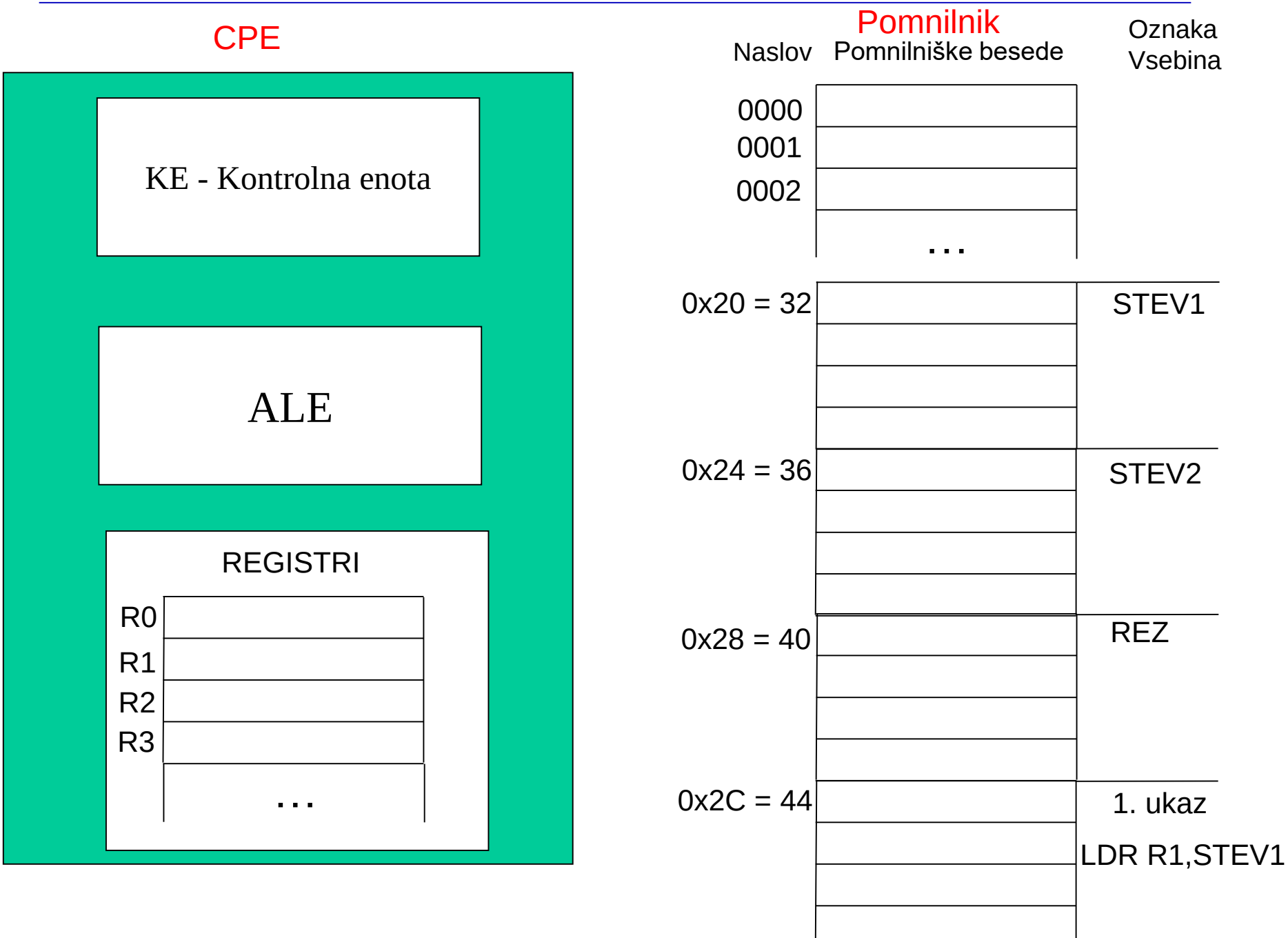
STEV1	64
STEV2	16
REZ	80

Python 2.7

```
1  STEV1=0x40
2  STEV2=0x10
3  REZ = STEV1 + STEV2
4  print (" STEV1 = " + hex(STEV1) + "\n+STEV2 = " +
```

http://goo.gl/YXQ5qN

Zgled: izvedba programa za seštevanje dveh števil



Zgled: izvedba programa za seštevanje dveh števil

UKAZI

	Strojni jezik	Zbirni jezik	Opis ukaza	Komentar
1.	0xE51F1014	ldr r1, stev1	R1 ← M[0x20]	
2.	0xE51F2014	ldr r2, stev2	R2 ← M[0x24]	
3.	0xE0823001	add r3, r2, r1	R3 ← R1 + R2	
4.	0xE50F3018	str r3, rez	M[0x28] ← R3	

Pravilo tankega in debelega konca / Big vs. Little Endian

MSB LSB

0 x AA BB CC DD

Debeli konec
Big Endian

Tanki konec
Little Endian