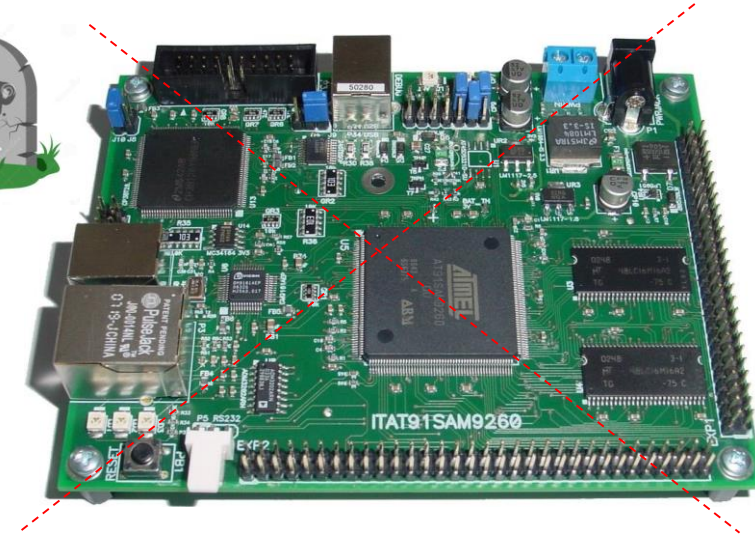


Računalniška arhitektura RA

Računalnik STM32H750-DK



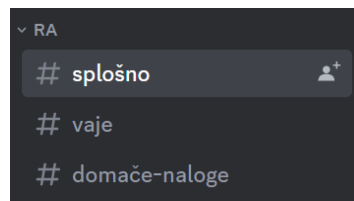
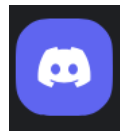
- Računalnik FRI-SMS
 - Mikrokmlnik AT91SAM9260 iz družine mikrokmlnikov ARM9



Ekpa RA Tutorji



Žiga Pušnik
[ziga.pusnik@fri....](mailto:ziga.pusnik@fri...)



<https://discord.com/channels/110837568099713027/11156262933416902757>



Andrej Sušnik
as1767@student.uni-lj.si



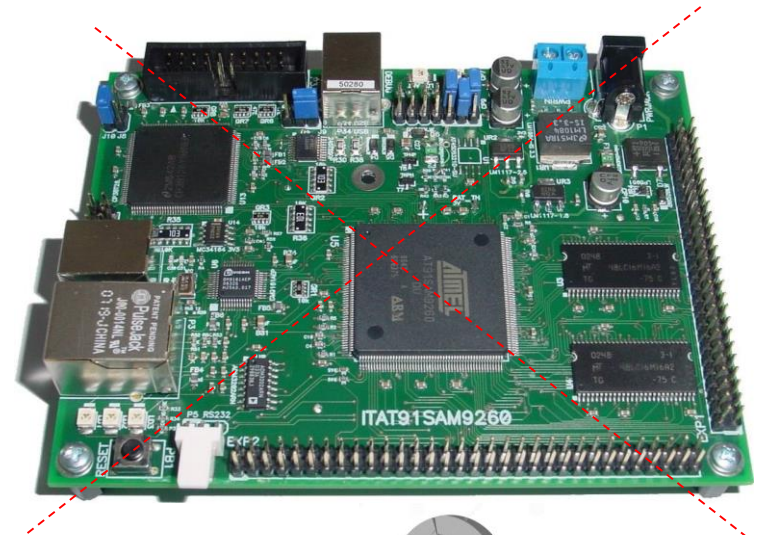
Robert Rozman
rozman@fri.uni-lj.si

Računalniška arhitektura RA

Računalnik STM32H750-DK



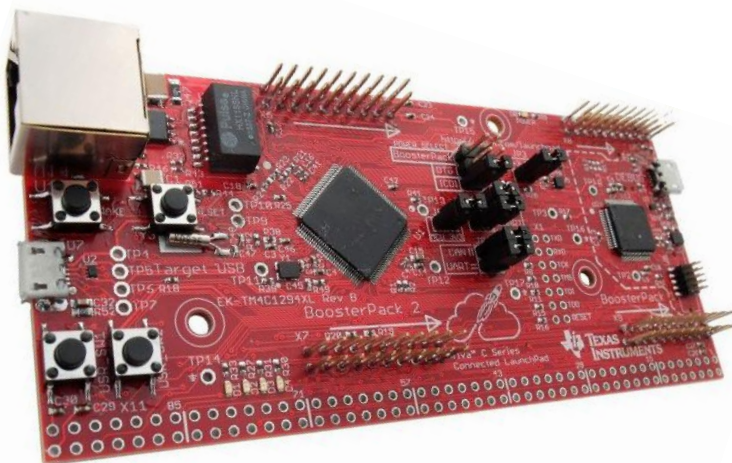
- Računalnik FRI-SMS
 - Mikrokontroler AT91SAM9260 iz družine mikrokontrolerov ARM9



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 (2. praktični vaji) + domača naloga
- Tri preverjanja (november, december, januar)
- Priprava na izpit (avditorne naloge)
- Predmetni seminar po dogovoru z asistentom

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*

Spletni simulator cpulator

- <https://cpulator.01xz.net/?sys=arm>
- začetni projekt RA:
 - <https://cpulator.01xz.net/?sys=arm&loadasm=share/sg8LlNt.s>

The screenshot displays the cpulator web interface with the following components:

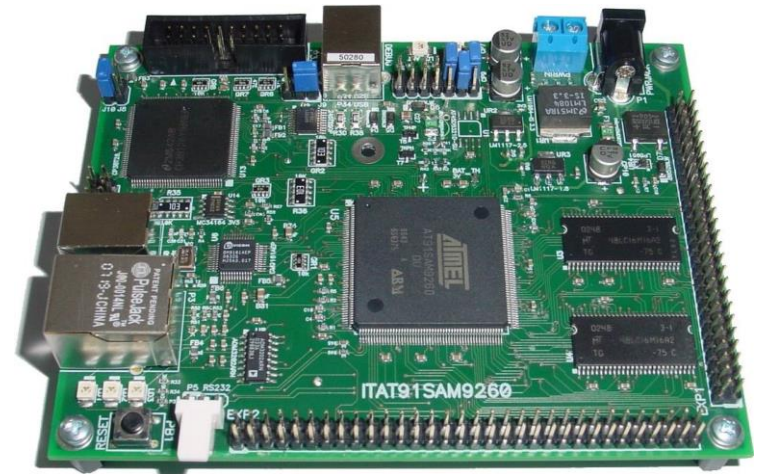
- Control Bar:** Stopped, Step Into (F2), Step Over (Ctrl-F2), Step Out (Shift-F2), Continue (F3), Stop (F4), Restart (Ctrl-R), Reload (Ctrl-Shift-L), File, Help.
- Registers Panel:** Lists registers r0-r12, sp, lr, and pc. The pc register is highlighted with the value 00000048.
- Editor (Ctrl-E):** Shows assembly code for 'untitled.s' in ARMv7. The code includes directives like .text, .org, @spremenljivke, .word, .space, .align, .global, and instructions like adr, ldr, add, and str.
- Memory (Ctrl-M):** Shows a memory dump starting at address 00000040, with contents of 'aa aa aa aa' repeated.
- Messages Panel:** Shows the compilation process: 'Compiling...', 'Code and data loaded from ELF executable into memory. Total size is 80 bytes.', and 'Compile succeeded.'

Računalniška arhitektura RA

Računalnik STM32H750-DK



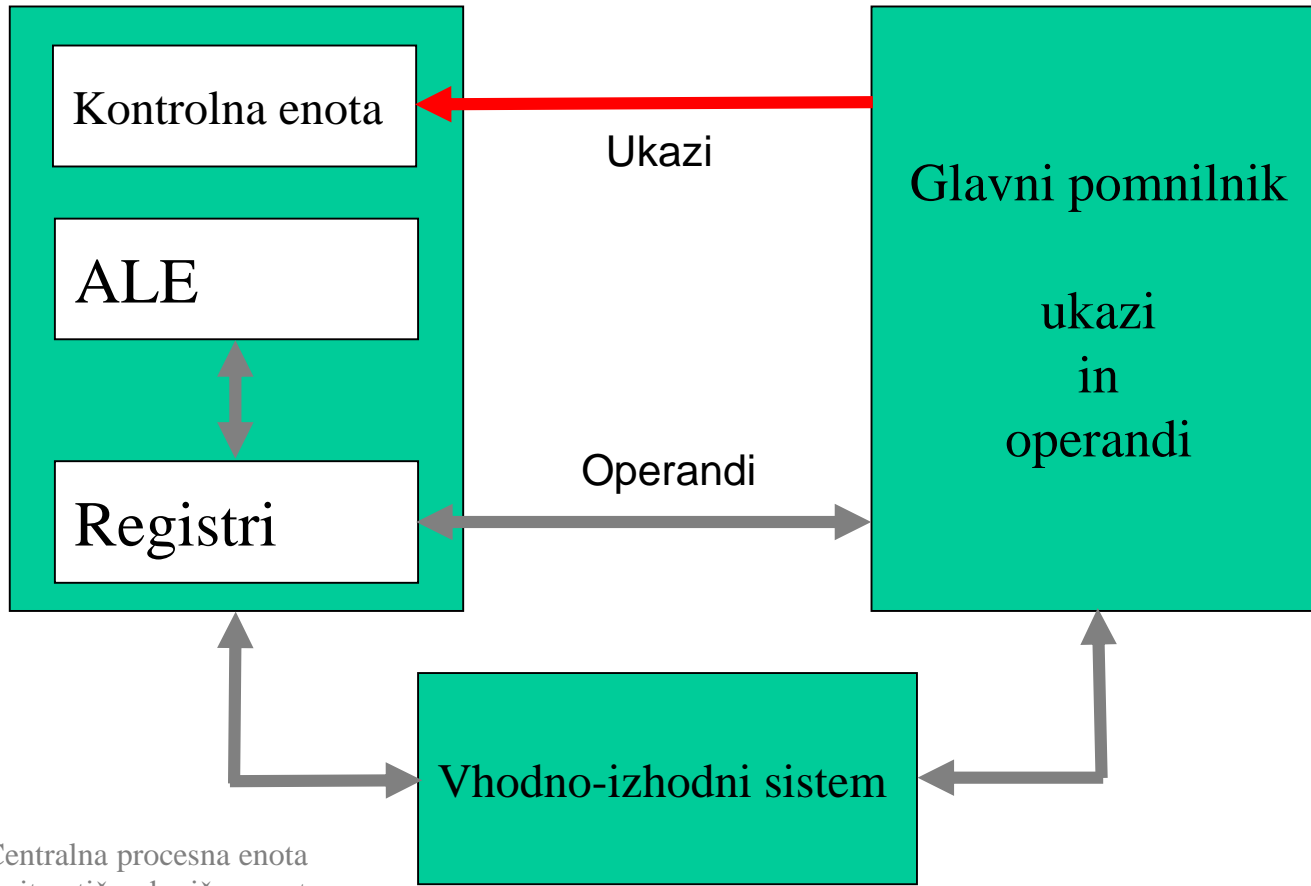
- Računalnik FRI-SMS
 - Mikrokontroler AT91SAM9260 iz družine mikrokontrolerov ARM9



LAB 1.2 Von Neumannov model (VN)

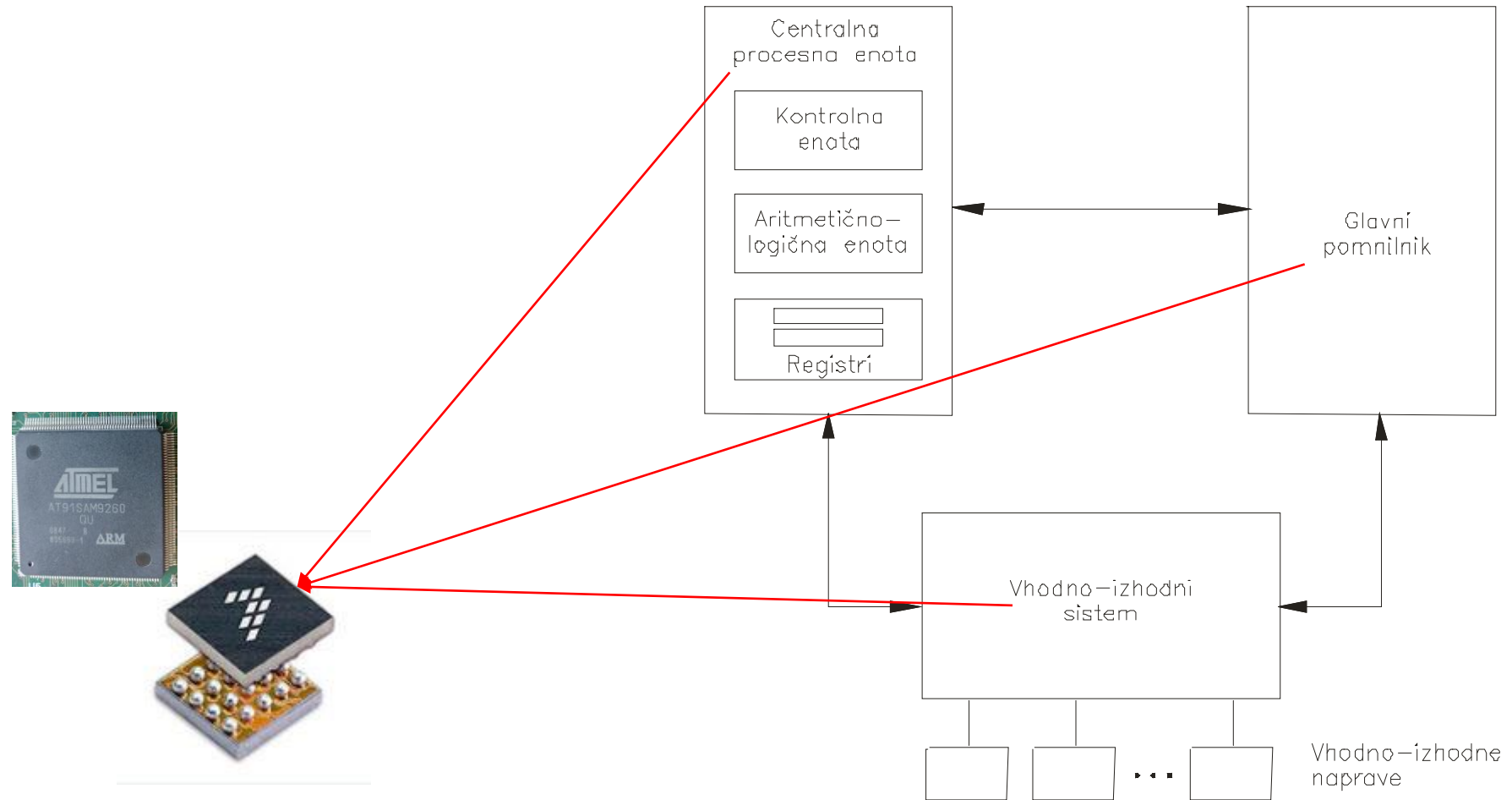
Von Neumannov računalniški model

CPE



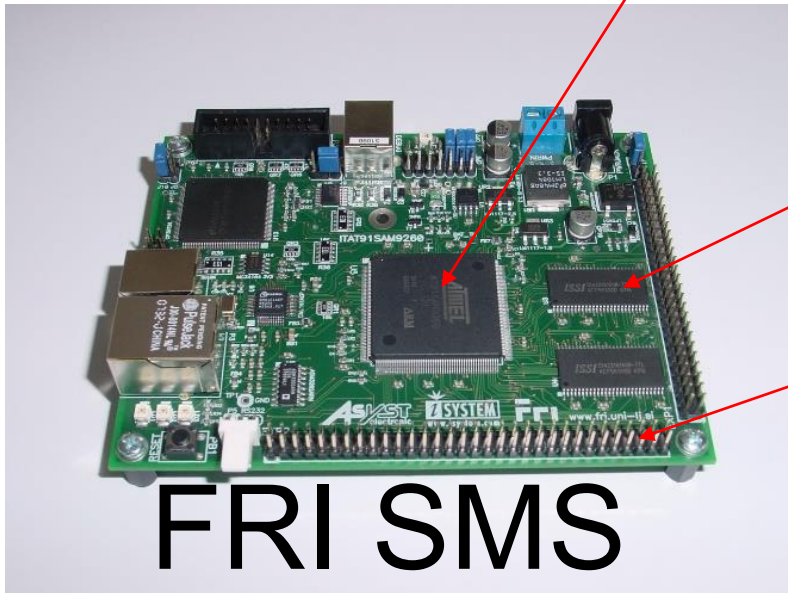
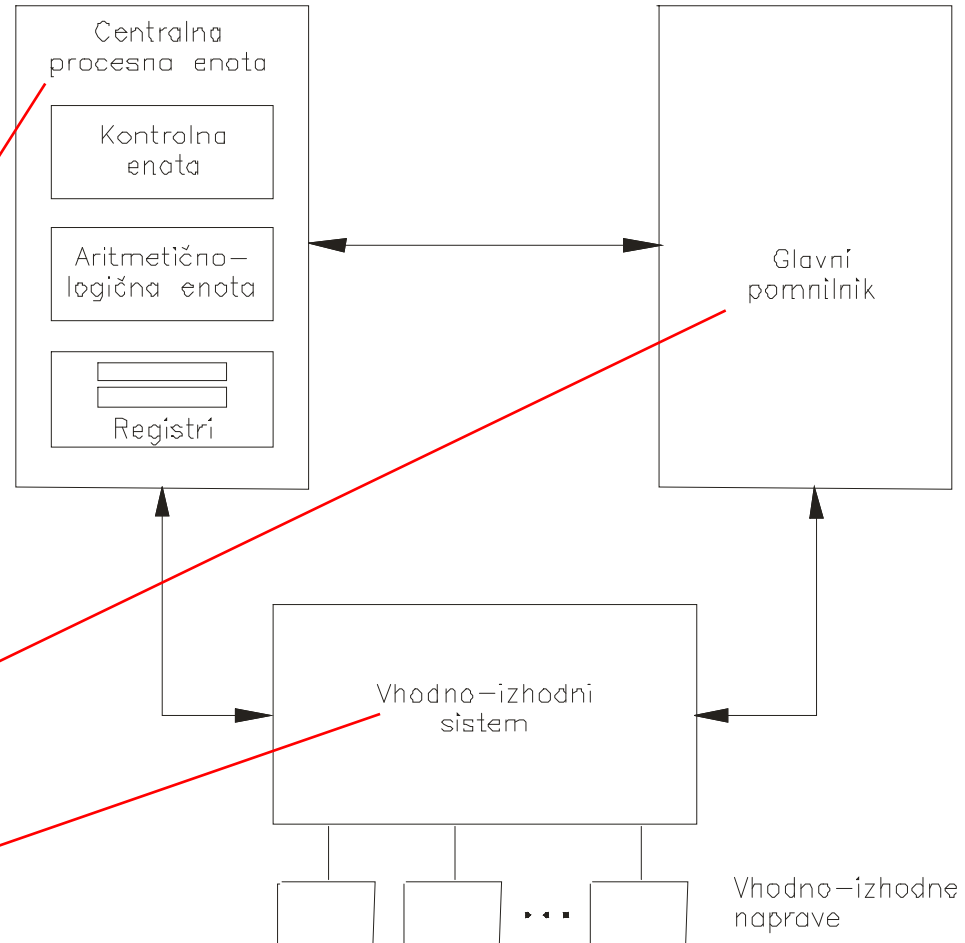
CPE – Centralna procesna enota
ALE – Aritmetično logična enota

Osnovni model računalnika



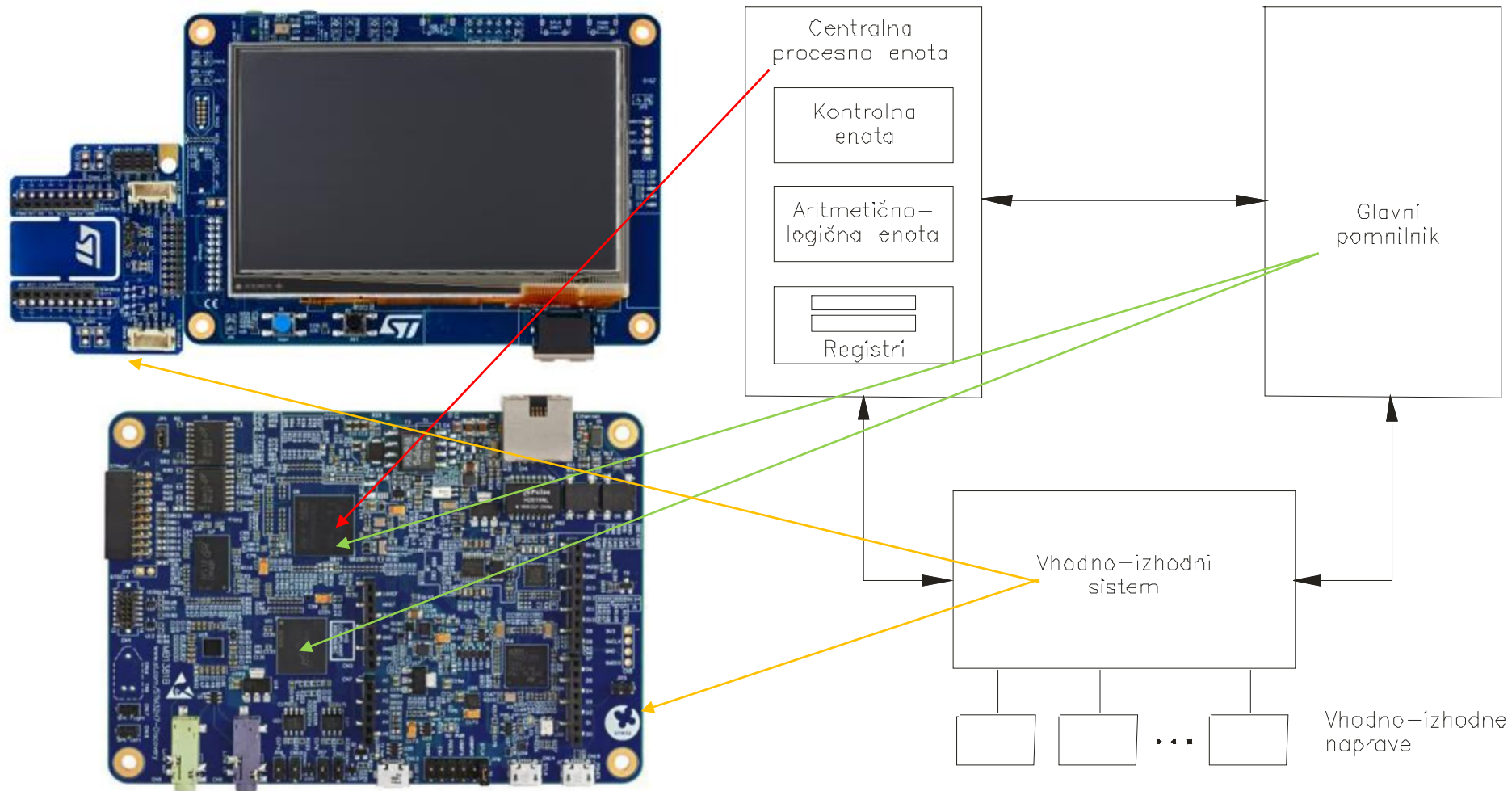
Mikrokontrolniki

Osnovni model računalnika



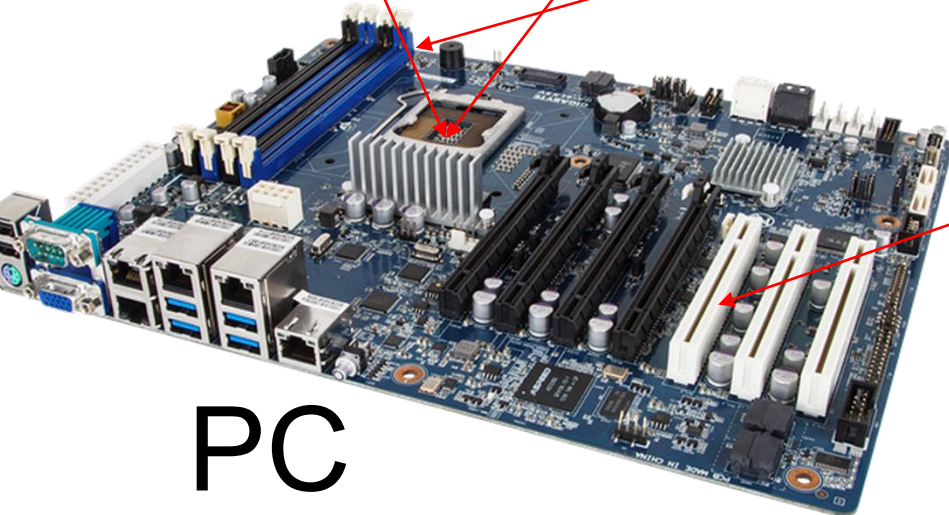
FRI SMS

Osnovni model računalnika

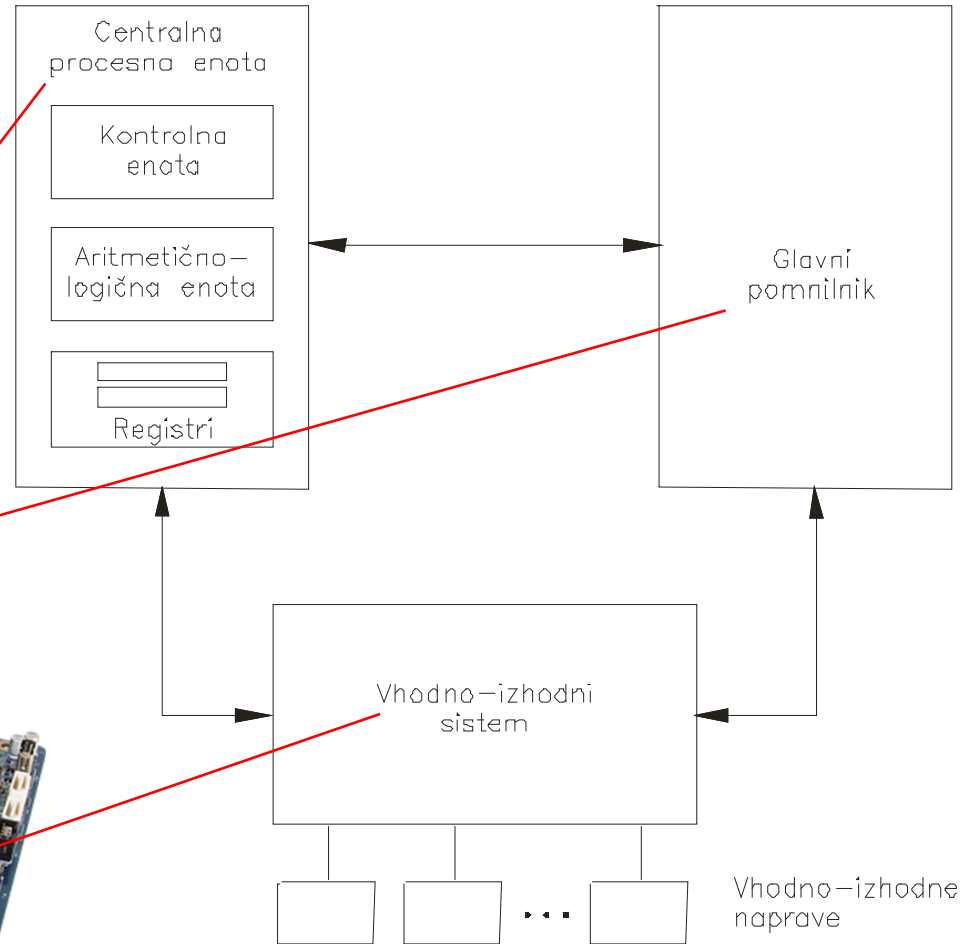


STM32H750-DK

Osnovni model računalnika



PC

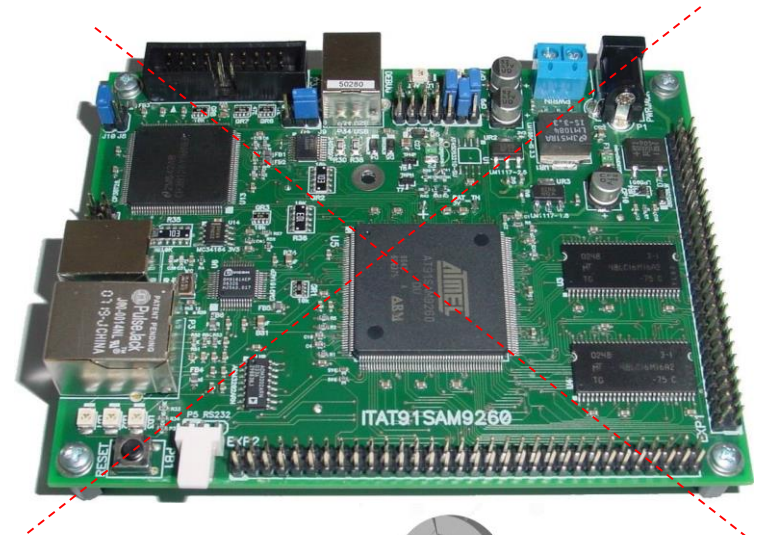


Računalniška arhitektura RA

Računalnik STM32H750-DK

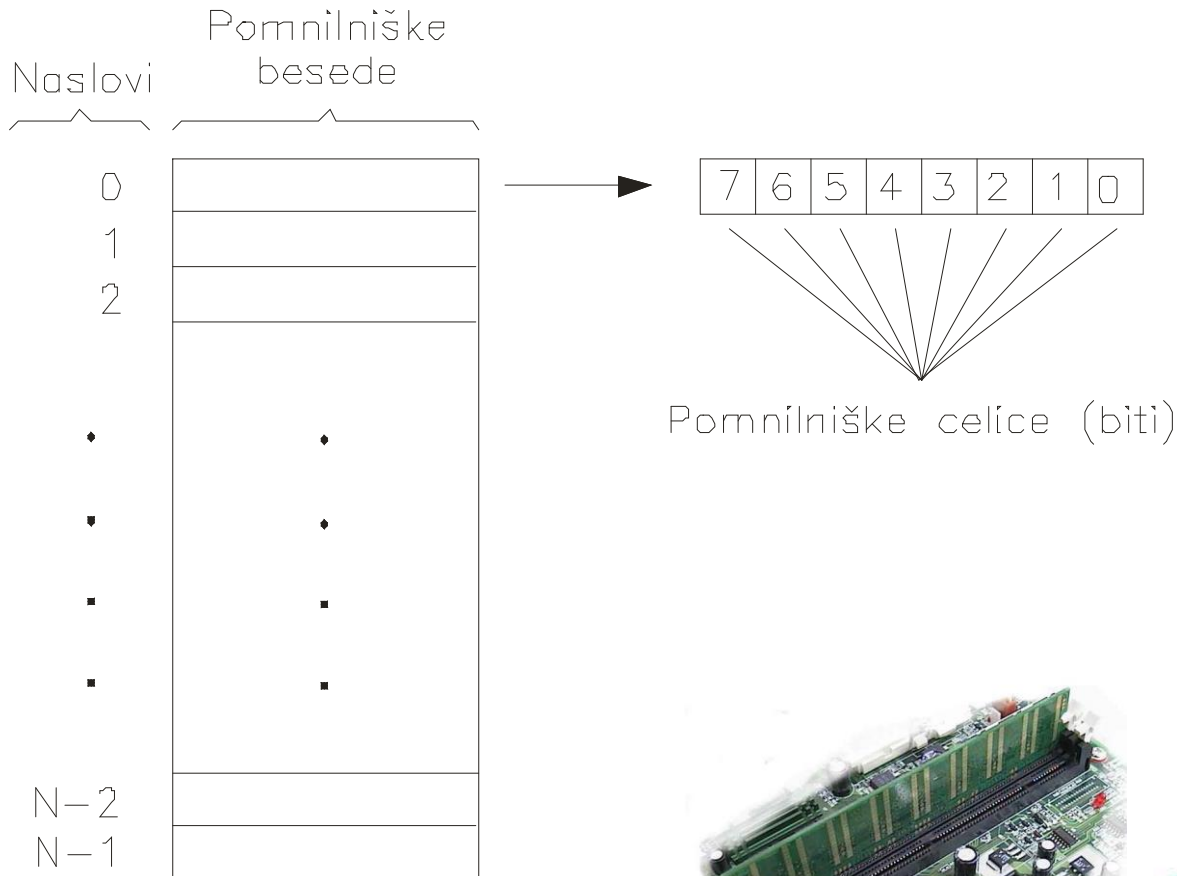


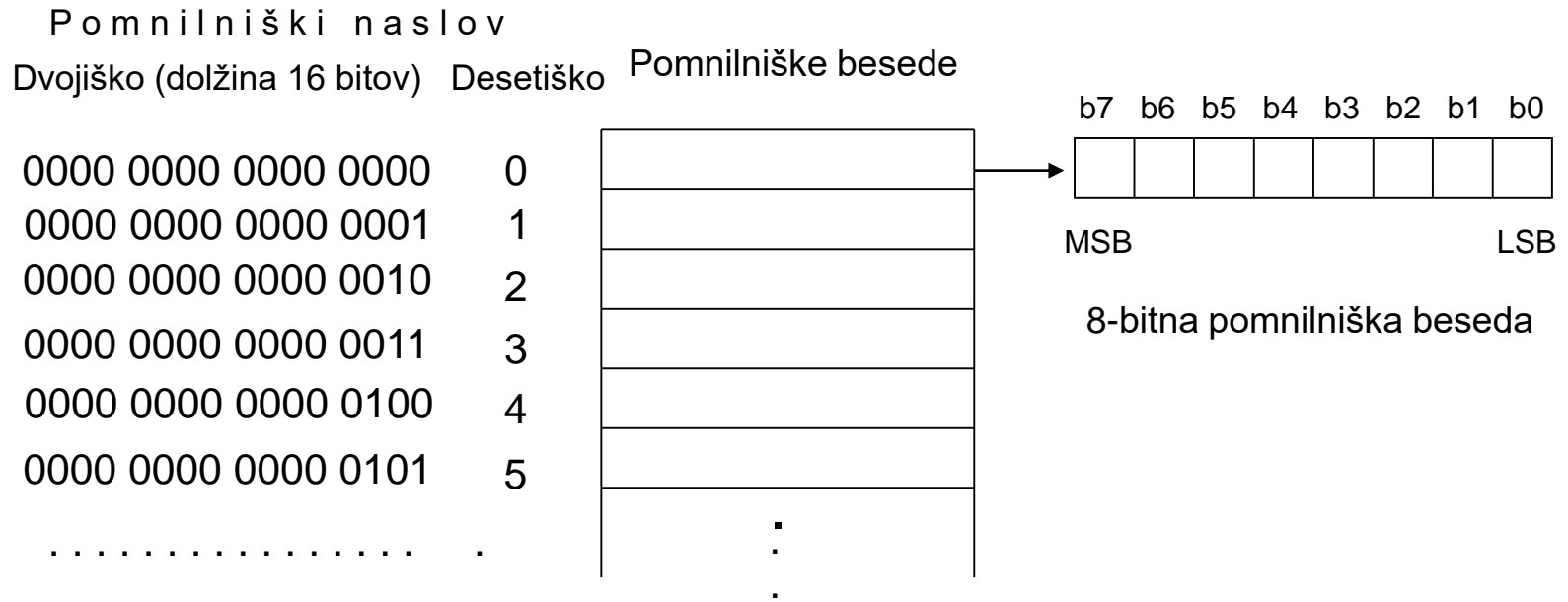
- Računalnik FRI-SMS
 - Mikrokontroler AT91SAM9260 iz družine mikrokontrolerov ARM9



LAB 1.3 Pomnilnik

Kaj je pomnilnik ?





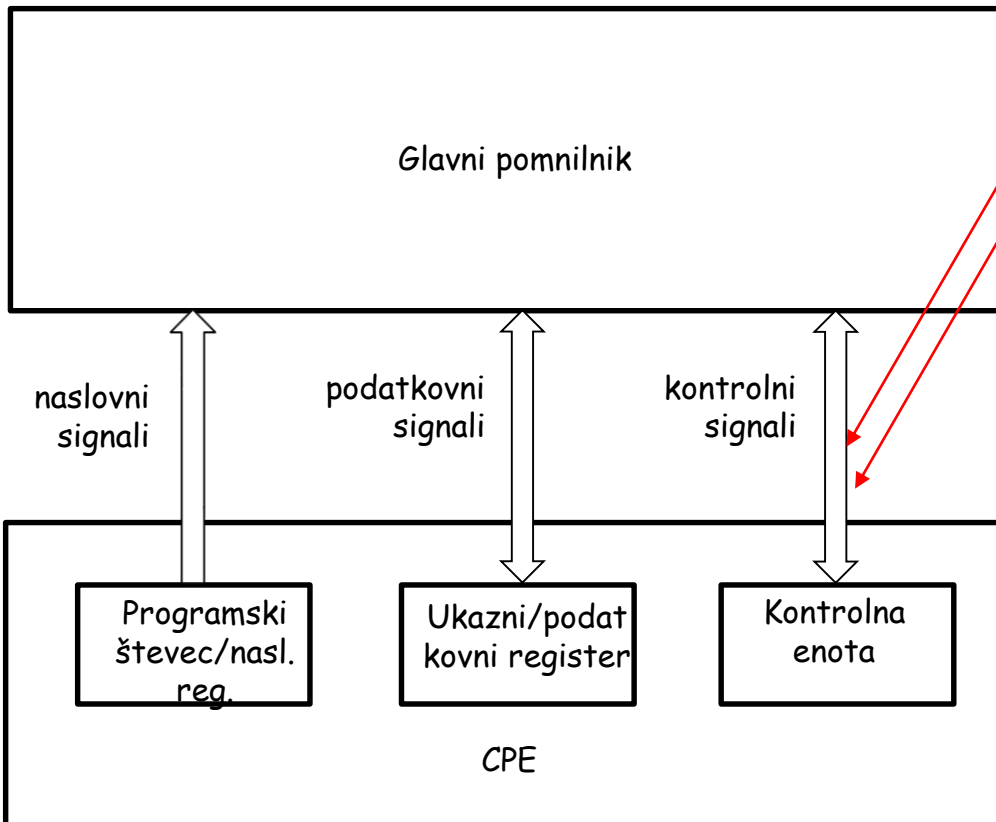
Pomnilniški naslov

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

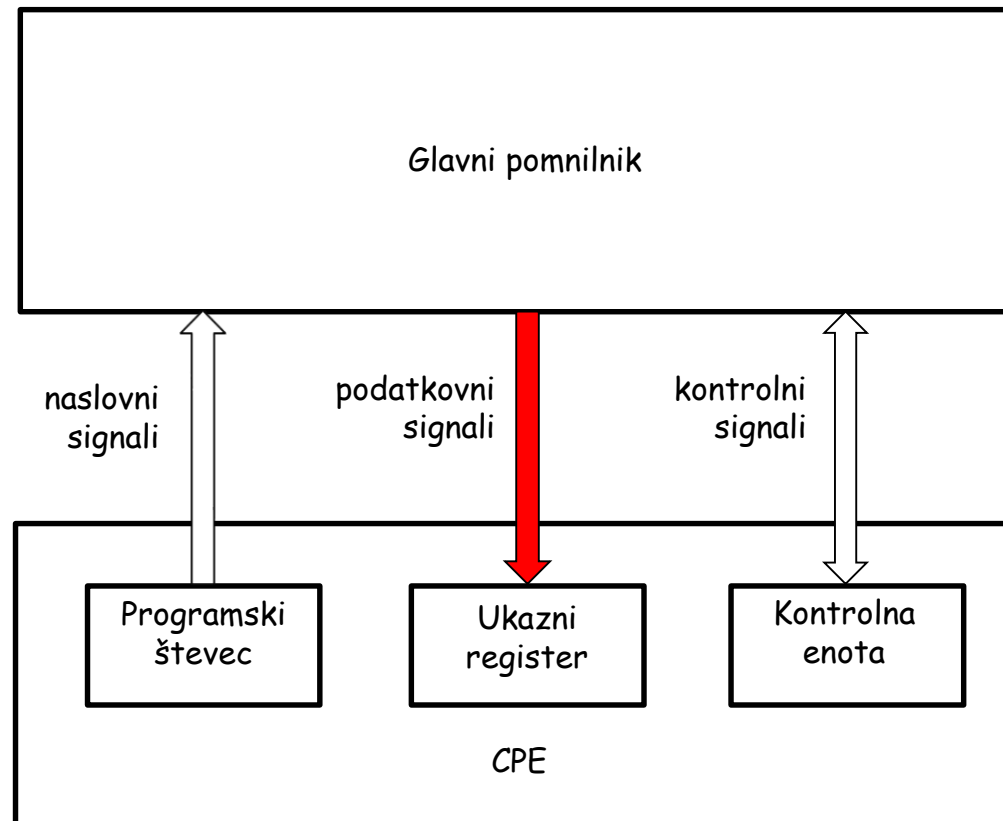
Vodilo = skupina povezav
(naslovno, podatkovno,
kontrolno, ...)

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



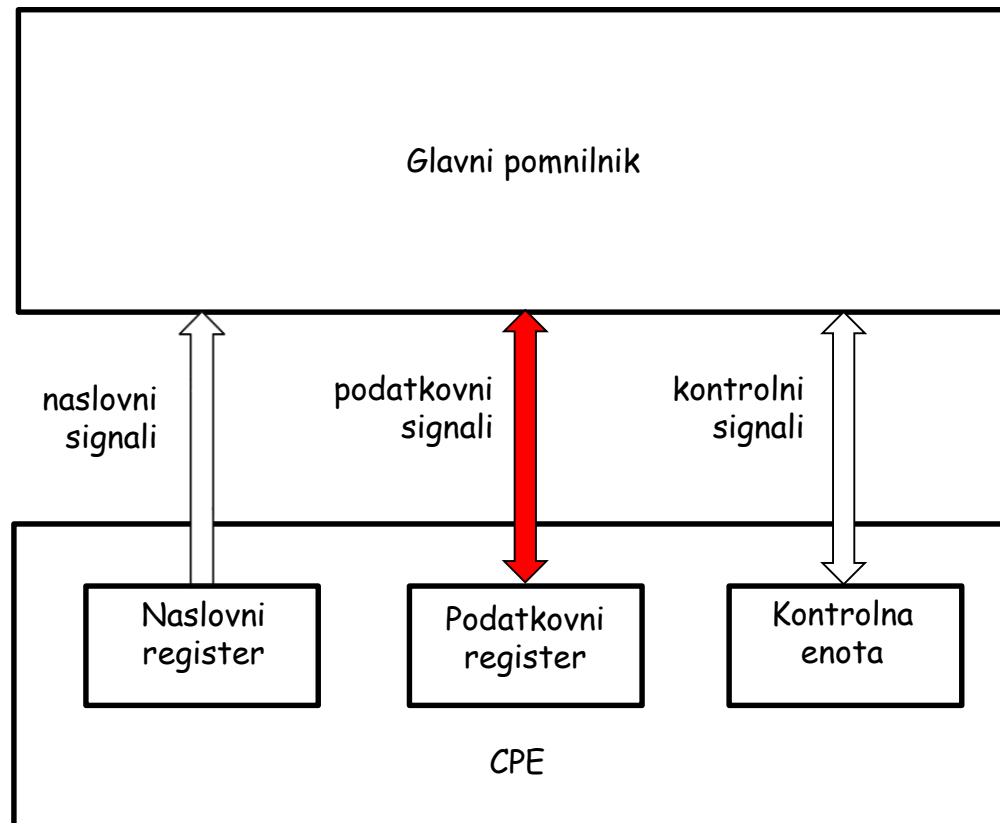
Kako CPE dostopa do glavnega pomnilnika?

Primer za ukaze:

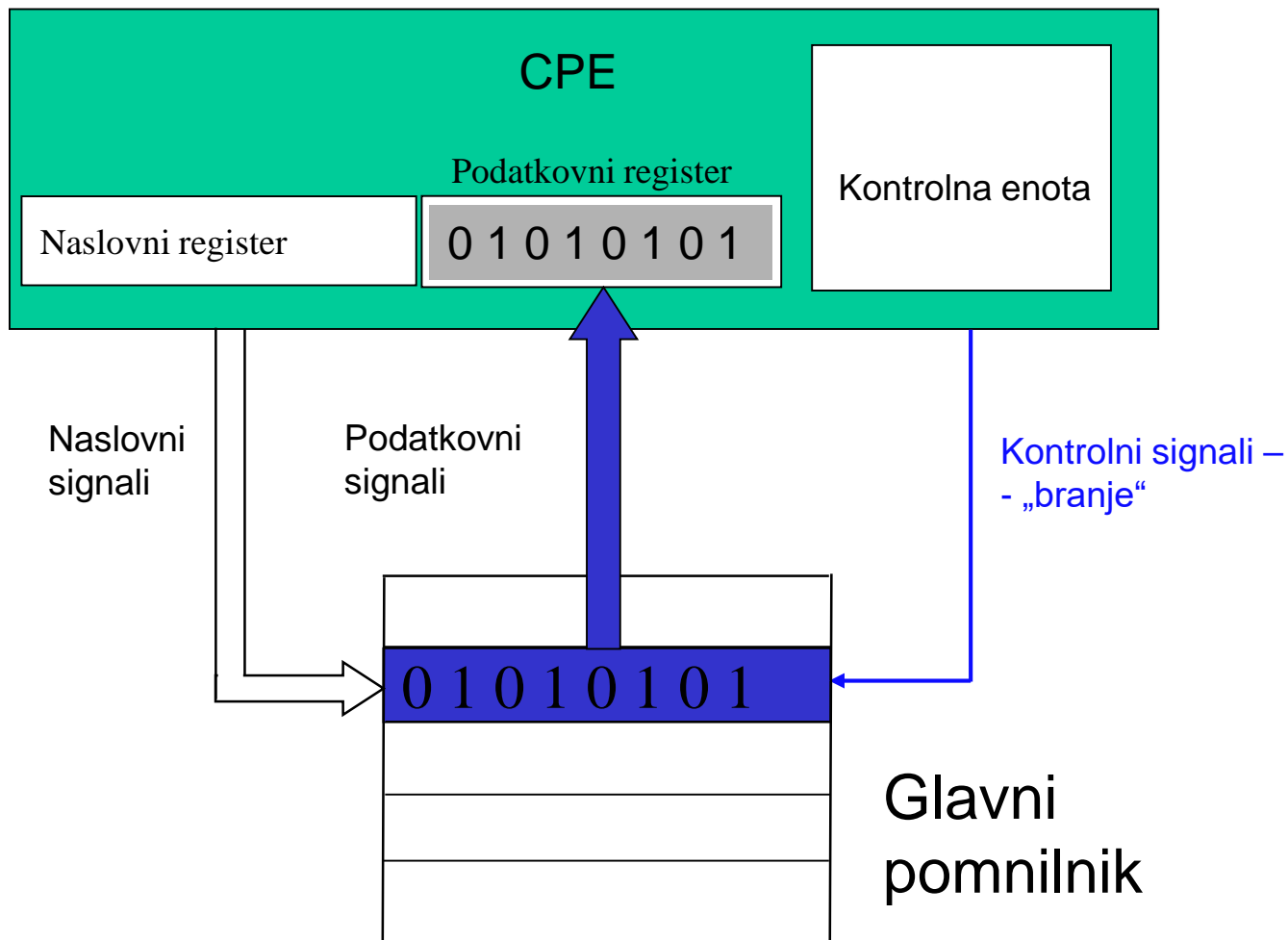


Kako CPE dostopa do glavnega pomnilnika?

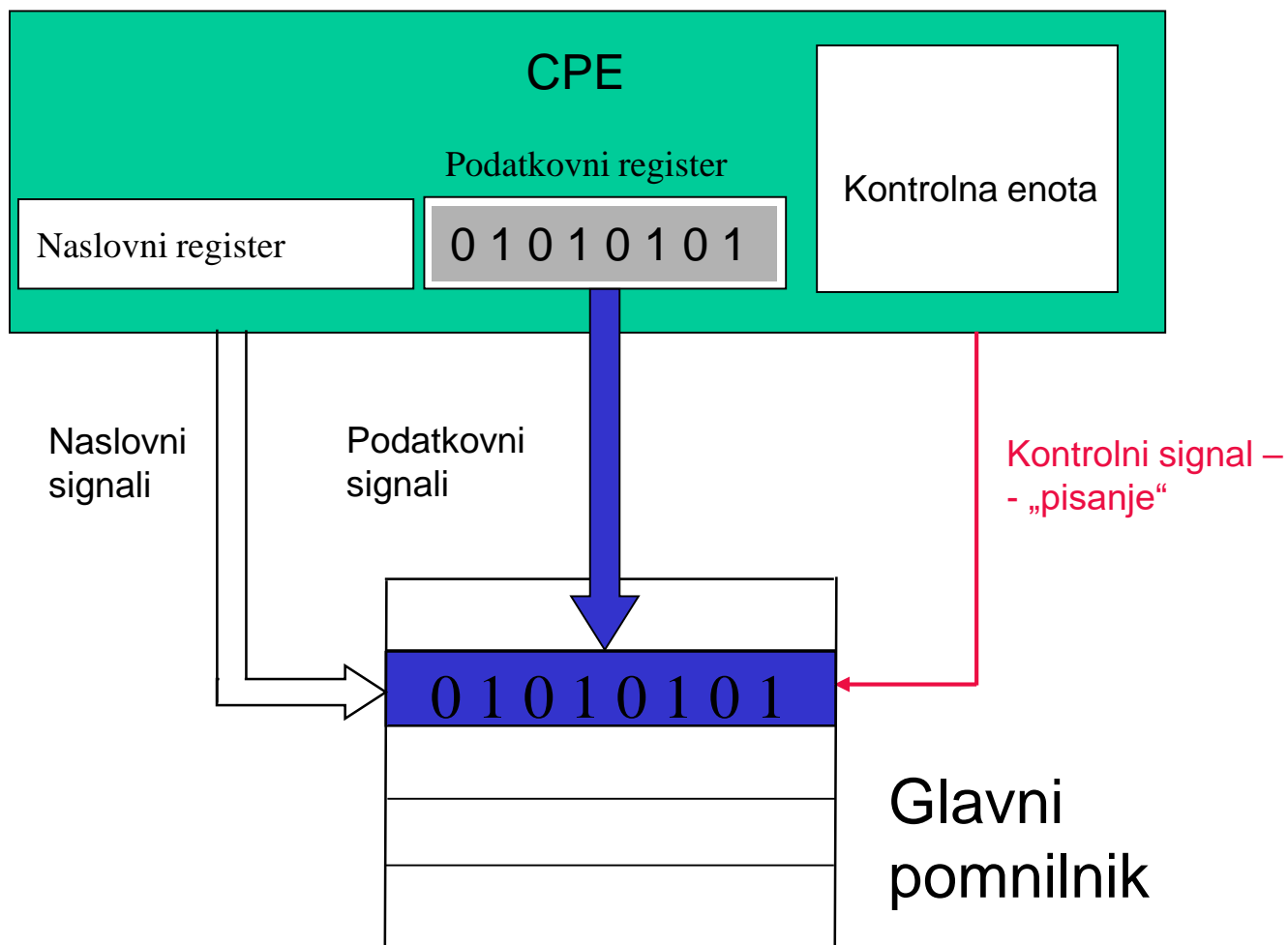
Primeri za operande:



Povezava med CPE in glavnim pomnilnikom – bralni dostop



Povezava med CPE in glavnim pomnilnikom – pisalni dostop

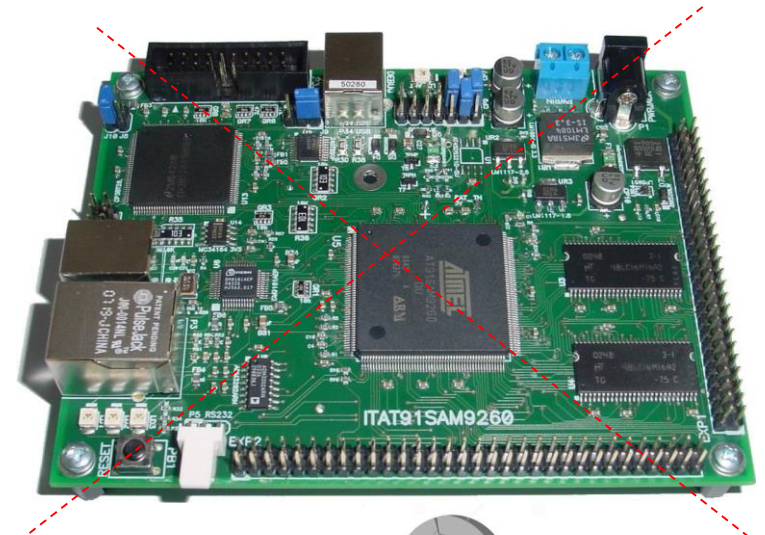


Računalniška arhitektura RA

Računalnik STM32H750-DK



- Računalnik FRI-SMS
 - Mikrokontrolnik AT91SAM9260 iz družine mikrokontrolnikov ARM9



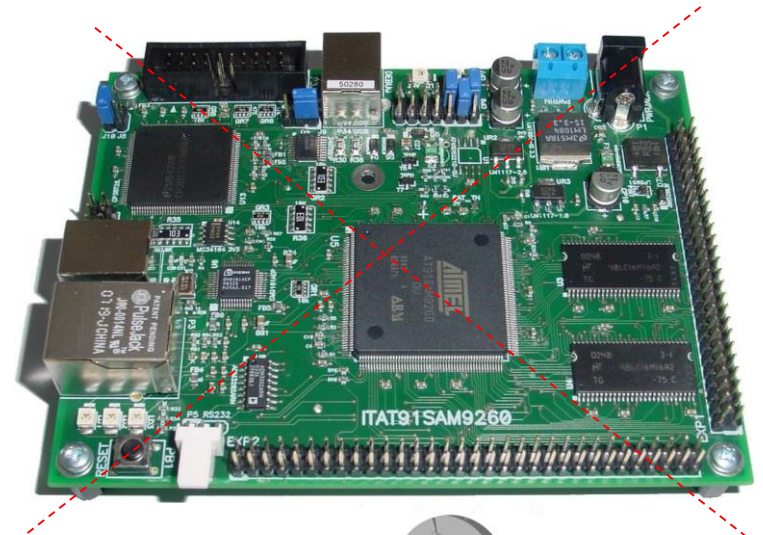
LAB 1.4 Številski sistemi (BIN,HEX) na hitro

Računalniška arhitektura RA

Računalnik STM32H750-DK



- Računalnik FRI-SMS
 - Mikrokontrolnik AT91SAM9260 iz družine mikrokontrolnikov ARM9



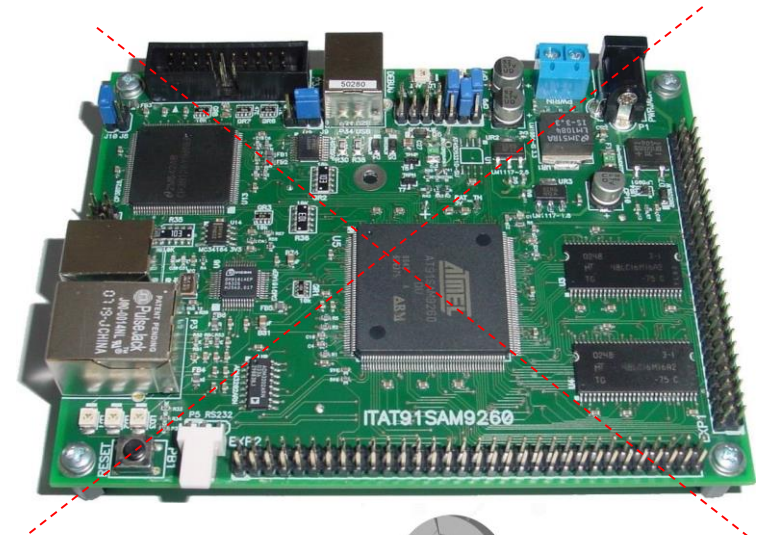
LAB 1.5 Pravilo tankega/debelega konca na hitro

Računalniška arhitektura RA

Računalnik STM32H750-DK

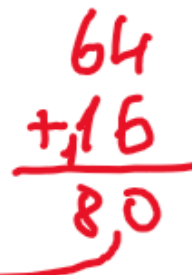


- Računalnik FRI-SMS
 - Mikrokontroler AT91SAM9260 iz družine mikrokontrolerov ARM9



LAB 1.6 Seštevanje – človek, python, zbirnik

Človek (zgled: $64 + 16 = 80$)

$$64 + 16 = ?$$


A handwritten red arrow points from the result '80' of the vertical addition to the question mark in the equation '64 + 16 = ?'.

$$\begin{array}{r} 64 \\ + 16 \\ \hline 80 \end{array}$$

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

Print output (drag lower right corner to resize)

Global frame

STEV1	64
STEV2	16
REZ	80

```
STEV1 = 0x40
```

```
+STEV2 = 0x10
```

```
-----
```

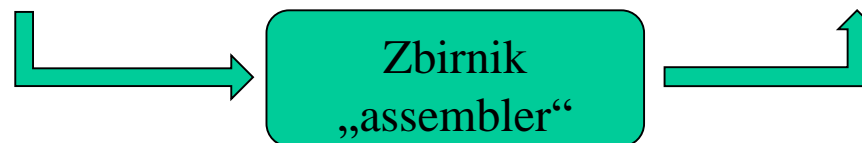
```
REZ = 0x50
```

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
adr r0, stev1	$R0 \leftarrow \text{nasl. stev1}$	0xE24F0014
ldr r1, [r0]	$R1 \leftarrow M[R0]$	0xE5901000
adr r0, stev2	$R0 \leftarrow \text{nasl. stev2}$	0xE24F0018
ldr r2, [r0]	$R2 \leftarrow M[R0]$	0xE5902000
add r3, r2, r1	$R3 \leftarrow R1 + R2$	0xE0823001
adr r0, rez	$R0 \leftarrow \text{nasl. rez}$	0xE24F0020
str r3, [r0]	$M[R0] \leftarrow R3$	0xE5803000

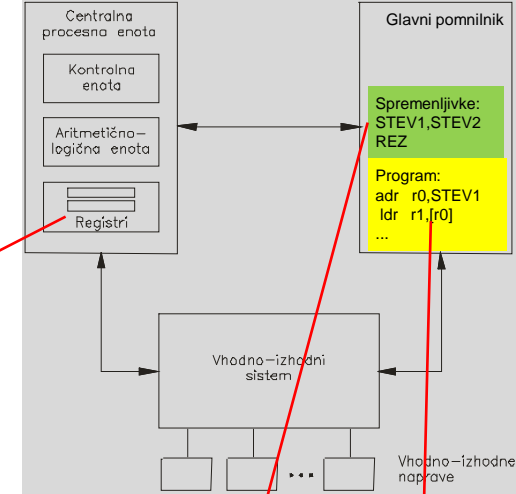


Ukaze izvajajte po korakih in opazujte vrednosti registrov in vrednosti spremenljivk v pomnilniku.

Praktično delo: vsota dveh števil

<https://cpulator.01xz.net/?sys=arm&loadasm=share/s8zU3xx.s>

Zbirni jezik	Opis ukaza	Strojni jezik
adr r0, stev1	R0 ← nasl. stev1	0xE24F0014
ldr r1, [r0]	R1 ← M[R0]	0xE5901000
adr r0, stev2	R0 ← nasl. stev2	0xE24F0018
ldr r2, [r0]	R2 ← M[R0]	0xE5902000
add r3, r2, r1	R3 ← R1 + R2	0xE0823001
adr r0, rez	R0 ← nasl. rez	0xE24F0020
str r3, [r0]	M[R0] ← R3	0xE5803000



Stopped

Step Into (F2) | Step Over (Ctrl-F2) | Step Out (Shift-F2) | Continue (F3) | Stop (F4) | Restart (Ctrl-R) | Reload (Ctrl-Shift-L) | File | Help

Registers

Refresh

r0	00000000
r1	00000000
r2	00000000
r3	00000000
r4	00000000
r5	00000000
r6	00000000
r7	00000000
r8	00000000
r9	00000000
r10	00000000
r11	00000000
r12	00000000
sp	00000000
lr	00000000
pc	0000002c
cpsr	000001d3 NZCVI SVC
spsr	00000000 NZCVI ?

Disassembly (Ctrl-D)

Go to address, label, or register: 00000000

Address	Opcode	Disassembly
00000020	00000010	STEV1: andeq r0, r0, r0,
00000024	00000040	STEV2: andeq r0, r0, r0,
00000028	00000000	REZ: andeq r0, r0, r0
0000002c	e24f0014	14 adr r0, STEV1
00000030	e5901000	15 ldr r1, [r0]
00000034	e24f0018	17 adr r0, STEV2
00000038	e5902000	18 ldr r2, [r0]
0000003c	e0813002	20 add r3, r1, r2
00000040	e24f0020	22 adr r0, REZ
00000044	e5803000	23 str r3, [r0]
00000048	eaaffffe	26 end: b er end: b 0x48 (0x48: enc

Memory (Ctrl-M)

Go to address, label, or register:

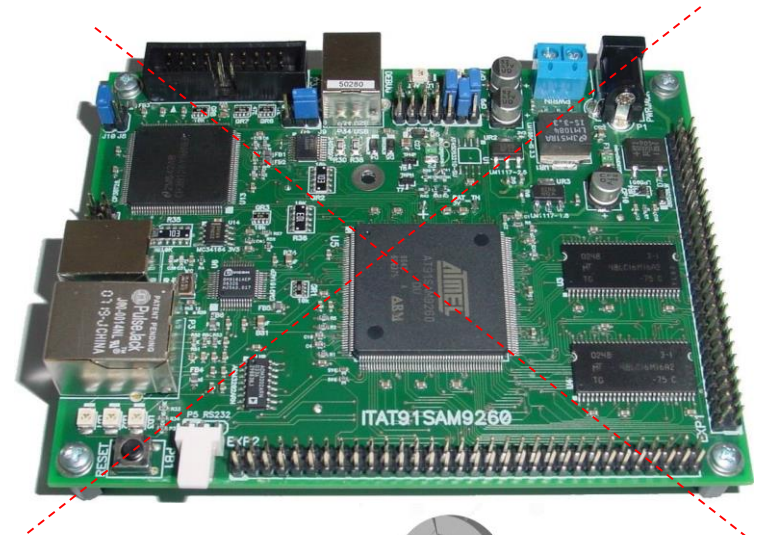
Address	Memory contents and ASCII
00000000	00000000 00000000 00000000 00000000
00000010	00000000 00000000 00000000 00000000
00000020	00000010 00000040 00000000 e24f0014
00000030	e5901000 e24f0018 e5902000 e0813002
00000040	e24f0020 e5803000 eaaffffe 00000000
00000050	aaaaaaaa aaaaaaaaa aaaaaaaaa aaaaaaaaa
00000060	aaaaaaaa aaaaaaaaa aaaaaaaaa aaaaaaaaa
00000070	aaaaaaaa aaaaaaaaa aaaaaaaaa aaaaaaaaa
00000080	aaaaaaaa aaaaaaaaa aaaaaaaaa aaaaaaaaa
00000090	aaaaaaaa aaaaaaaaa aaaaaaaaa aaaaaaaaa
000000a0	aaaaaaaa aaaaaaaaa aaaaaaaaa aaaaaaaaa
000000b0	aaaaaaaa aaaaaaaaa aaaaaaaaa aaaaaaaaa
000000c0	aaaaaaaa aaaaaaaaa aaaaaaaaa aaaaaaaaa
000000d0	aaaaaaaa aaaaaaaaa aaaaaaaaa aaaaaaaaa
000000e0	aaaaaaaa aaaaaaaaa aaaaaaaaa aaaaaaaaa
000000f0	aaaaaaaa aaaaaaaaa aaaaaaaaa aaaaaaaaa
00000100	aaaaaaaa aaaaaaaaa aaaaaaaaa aaaaaaaaa
00000110	aaaaaaaa aaaaaaaaa aaaaaaaaa aaaaaaaaa
00000120	aaaaaaaa aaaaaaaaa aaaaaaaaa aaaaaaaaa
00000130	aaaaaaaa aaaaaaaaa aaaaaaaaa aaaaaaaaa
00000140	aaaaaaaa aaaaaaaaa aaaaaaaaa aaaaaaaaa
00000150	aaaaaaaa aaaaaaaaa aaaaaaaaa aaaaaaaaa
00000160	aaaaaaaa aaaaaaaaa aaaaaaaaa aaaaaaaaa

Računalniška arhitektura RA

Računalnik STM32H750-DK



- Računalnik FRI-SMS
 - Mikrokontroler AT91SAM9260 iz družine mikrokontrolerov ARM9



LAB 1.7 Predloge za zapiske

Python (zgles: 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 = " + hex(STE
```

<http://goo.gl/YXQ5qN>

Zgled: izvedba programa za seštevanje dveh števil

CPE



Pomnilnik

Naslov	Pomnilniške besede	Oznaka Vsebina
0000		
0001		
0002		
	...	
0x20 = 32		STEV1
0x24 = 36		STEV2
0x28 = 40		REZ
0x2C = 44		1. ukaz
		ADR R0,STEV1

Zgled: izvedba programa za seštevanje dveh števil

UKAZI

	Strojni jezik	Zbirni jezik	Opis ukaza	Komentar
1.	0xE24F0014	adr r0, stev1	$R0 \leftarrow \text{nasl. stev1}$	
2.	0xE5901000	ldr r1, [r0]	$R1 \leftarrow M[R0]$	
3.	0xE24F0018	adr r0, stev2	$R0 \leftarrow \text{nasl. stev2}$	
4.	0xE5902000	ldr r2, [r0]	$R2 \leftarrow M[R0]$	
5.	0xE0823001	add r3, r2, r1	$R3 \leftarrow R1 + R2$	
6.	0xE24F0020	adr r0, rez	$R0 \leftarrow \text{nasl. rez}$	
7.	0xE5803000	str r3, [r0]	$M[R0] \leftarrow R3$	

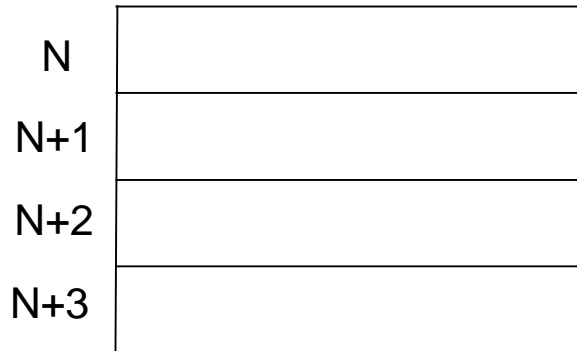
Pravilo tankega in debelega konca / Big vs. Little Endian

MSB

LSB

0 x AA BB CC DD

Debeli konec
Big Endian



Tanki konec
Little Endian

