

# Računalniška arhitektura RA

Računalnik STM32H750-DK



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



## Ekipa RA Tutorji



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



Anamari Orehar  
[a06477@student.unilj.si](mailto:a06477@student.unilj.si)



Kristian Šurbek  
[ks5453@student.unilj.si](mailto:ks5453@student.unilj.si)



Andrej Sušnik  
[as1767@student.uni-lj.si](mailto:as1767@student.uni-lj.si)



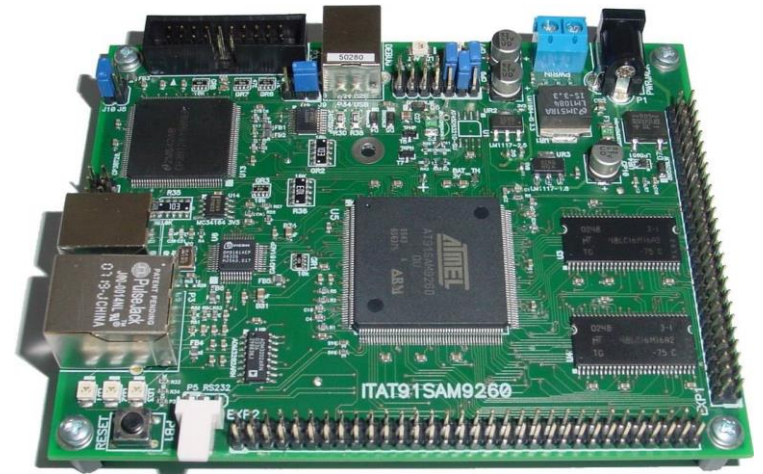
Robert Rozman  
[rozman@fri.uni-lj.si](mailto:rozman@fri.uni-lj.si)

# Računalniška arhitektura RA

Računalnik STM32H750-DK



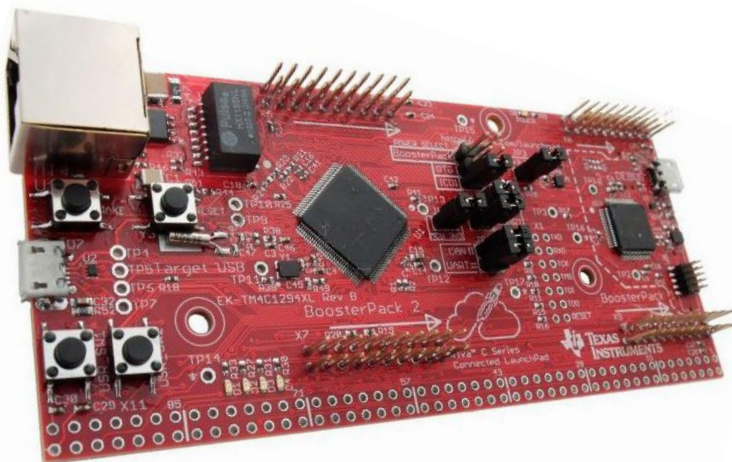
- Računalnik FRI-SMS
  - Mikrokontroler AT91SAM9260 iz družine mikrokontrolerov ARM9



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 + 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*

# Spletni simulator cpulator

- <https://cpulator.01xz.net/?sys=arm>

The screenshot displays the cpulator web interface. At the top, there are control buttons: Stopped, Step Into (F2), Step Over (Ctrl-F2), Step Out (Shift-F2), Continue (F3), Stop (F4), Restart (Ctrl-R), and Reload (Ctrl-Shift-L). There are also File and Help menus.

The main interface is divided into several panels:

- Registers:** A table showing registers r0 through r13 with their current values, all set to 00000000.
- Editor (Ctrl-E):** A code editor showing assembly code for ARMv7. The code includes labels for variables (stev1, stev2, rez), a loop, and instructions for loading, adding, and storing data.
- Memory (Ctrl-M):** A table showing memory addresses and their contents, which are mostly 'aa'.
- Messages:** A log showing the compilation process, including the assembly and linking commands used.

```
1 stev1: .word 0x40
2 stev2: .word 0x10
3 rez: .space 4
4
5 .global _start
6 _start:
7
8 adr r0, stev1
9 ldr r1, [r0]
10
11 adr r0, stev2
12 ldr r2, [r0]
13
14 add r3, r2, r1
15
16 adr r0, rez
17 str r3, [r0]
18
19 loop: b loop
20
21
```

Messages:

```
Code and data loaded from ELF-executable into memory. Total size is 48 bytes.
Assemble: arm-altera-eabi-as -mfloat-abi=soft -march=armv7-a -mcpu=cortex-a9 -mfpu=neon-fp16 --gdwarf2 -o work/asmV0FoCu.s.o work/a
Link: arm-altera-eabi-ld --script build_arm.ld -e _start -u _start -o work/asmV0FoCu.s.elf work/asmV0FoCu.s.o
Compile succeeded.
```

# 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

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	Registers
		<u>__start</u>	R0 00000000
		ldr r1, stev1	R1 00000000
000014101		ldr r1, [pc, -0014]	R2 00000000
		ldr r2, stev2	R3 00000000
000014201		ldr r2, [pc, -0014]	R4 00000000
		add r3, r2, r1	R5 00000000
00001308		add r3, r2, r1	R6 00000000
		str r3, rez	R7 00000000
000018300		str r3, [pc, -0018]	R8 00000000
		<u>end: b end</u>	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)

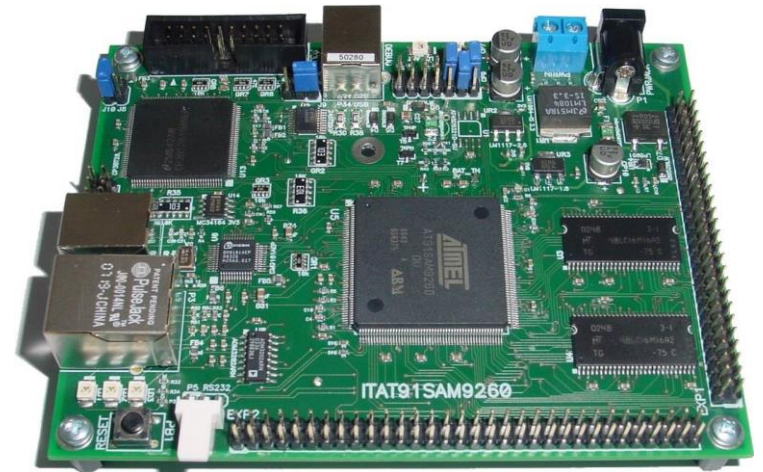
Build Find In Files Tools Script

# Računalniška arhitektura RA

Računalnik STM32H750-DK



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

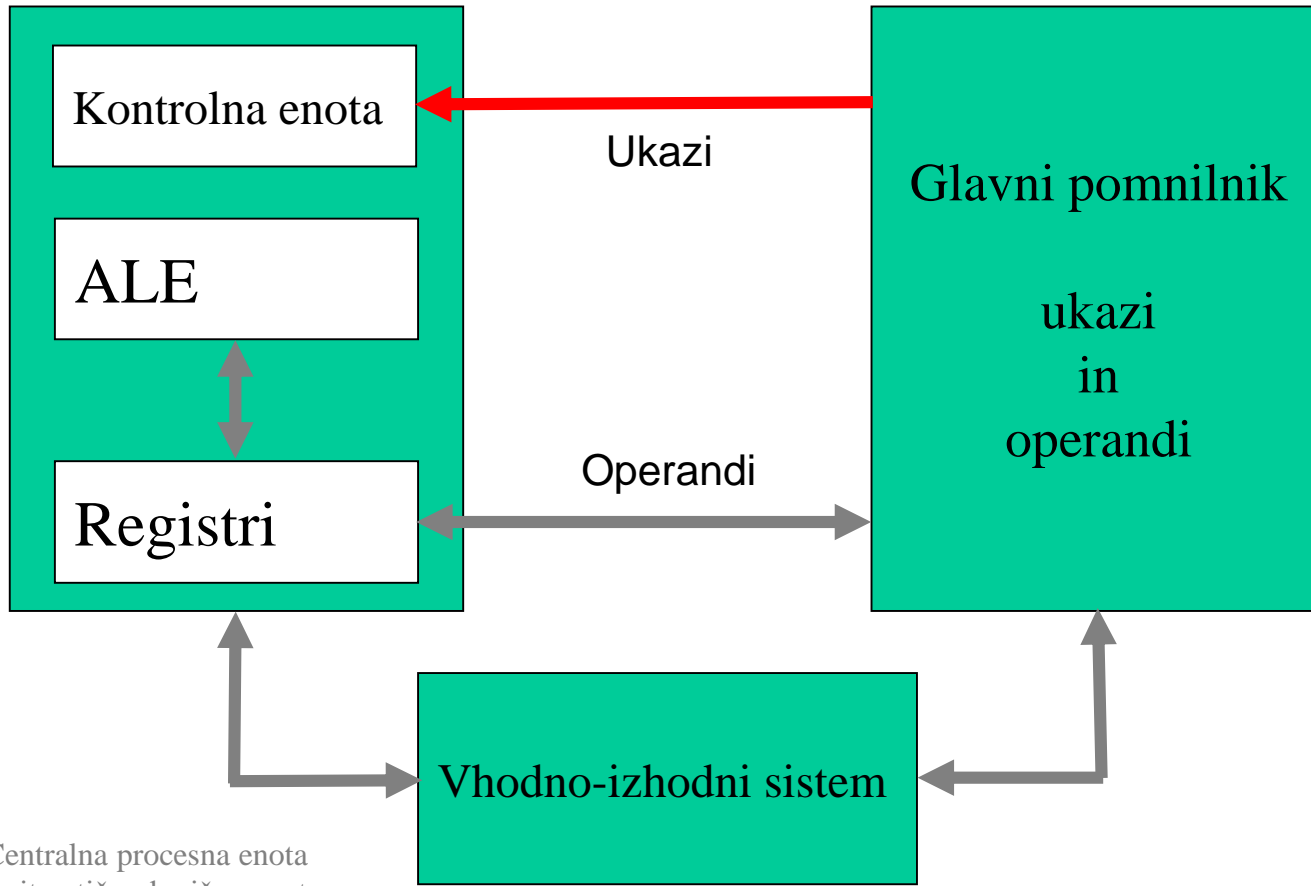


LAB 1.2 Von Neumannov model (VN)



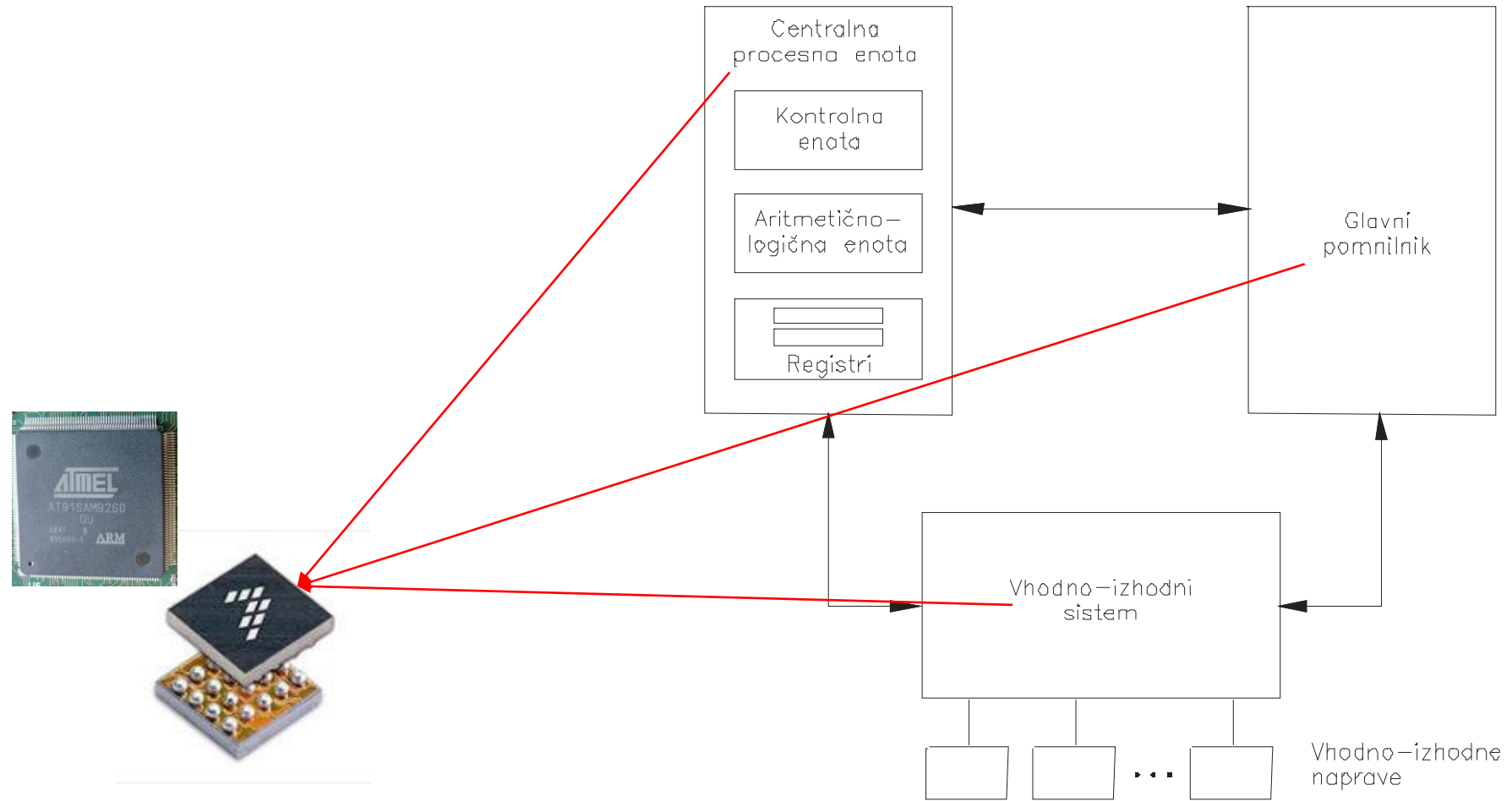
## Von Neumannov računalniški model

CPE



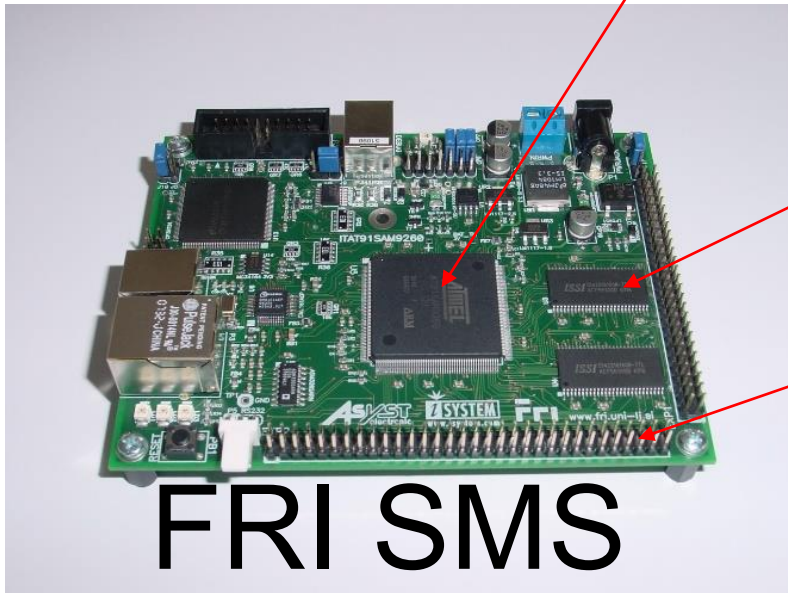
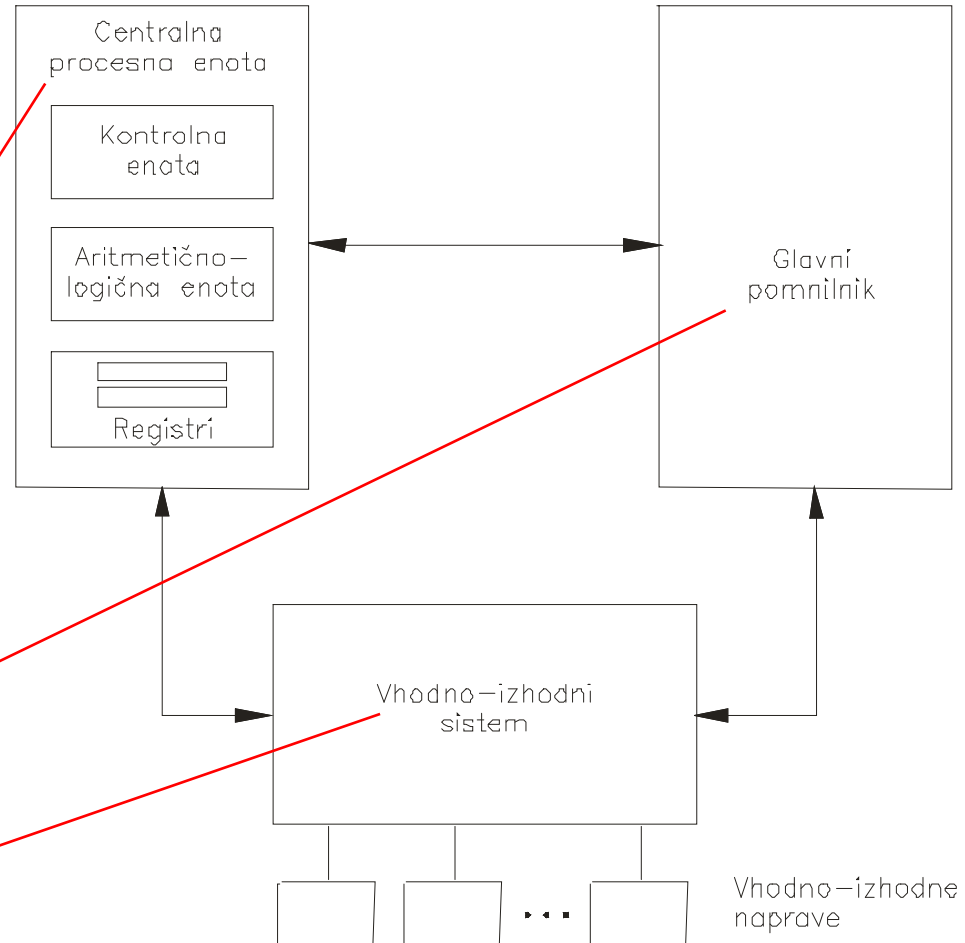
CPE – Centralna procesna enota  
ALE – Aritmetično logična enota

# Osnovni model računalnika



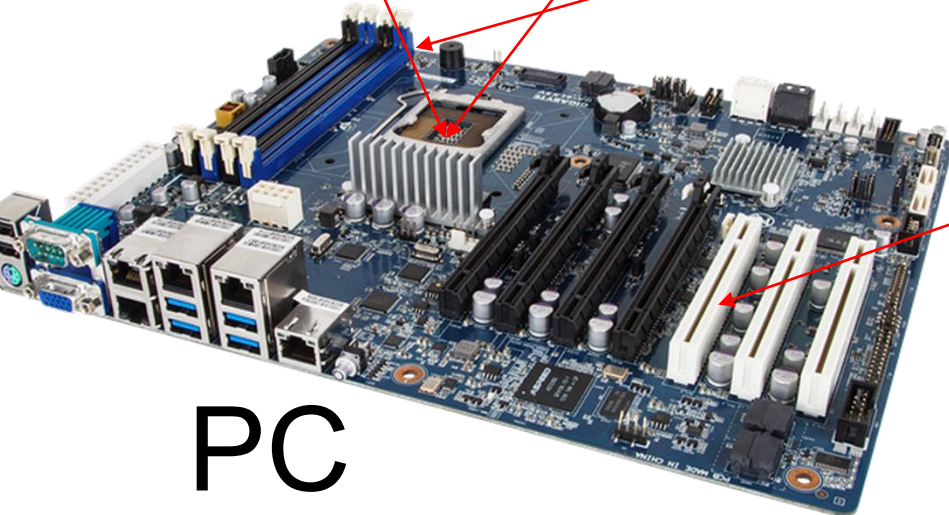
## Mikrokontrolniki

# Osnovni model računalnika

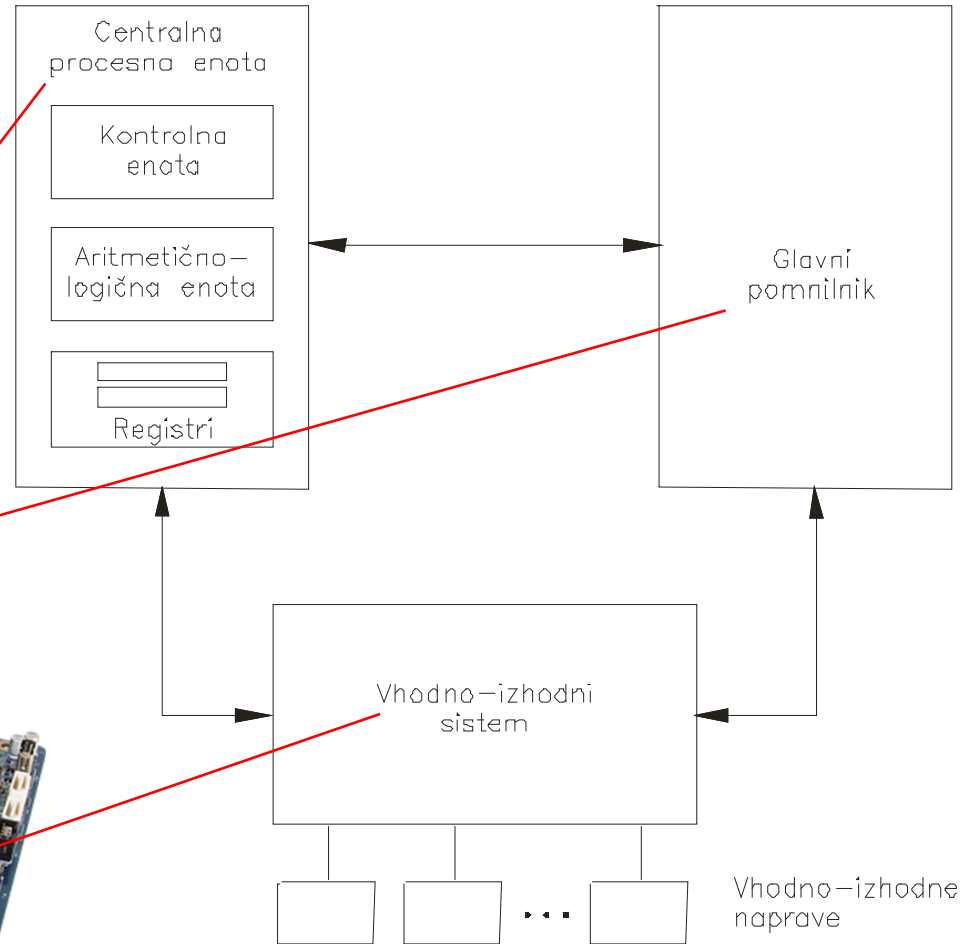


FRI SMS

# Osnovni model računalnika



PC



# Računalniška arhitektura RA

Računalnik STM32H750-DK

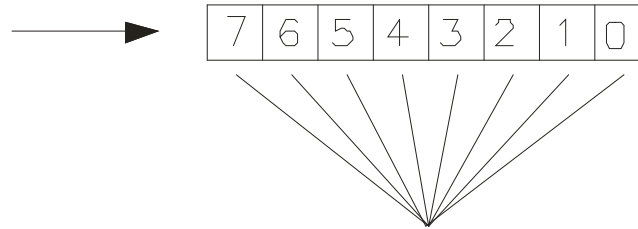
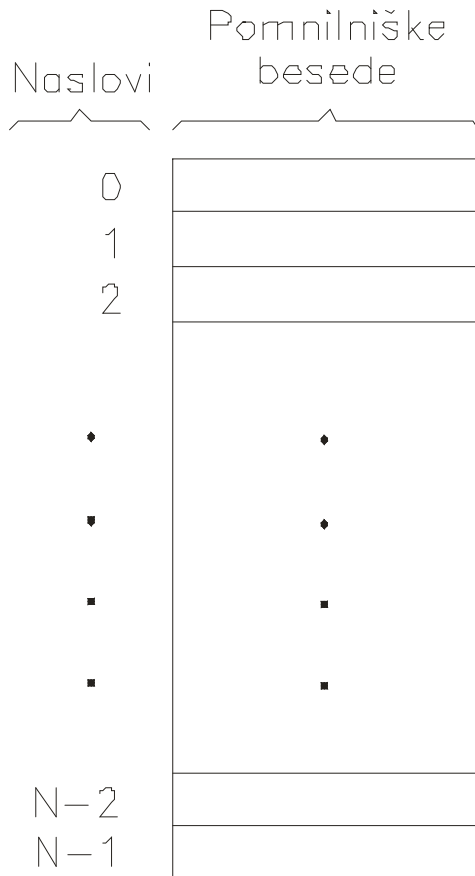


- Računalnik FRI-SMS
  - Mikrokontrolnik AT91SAM9260 iz družine mikrokontrolnikov ARM9



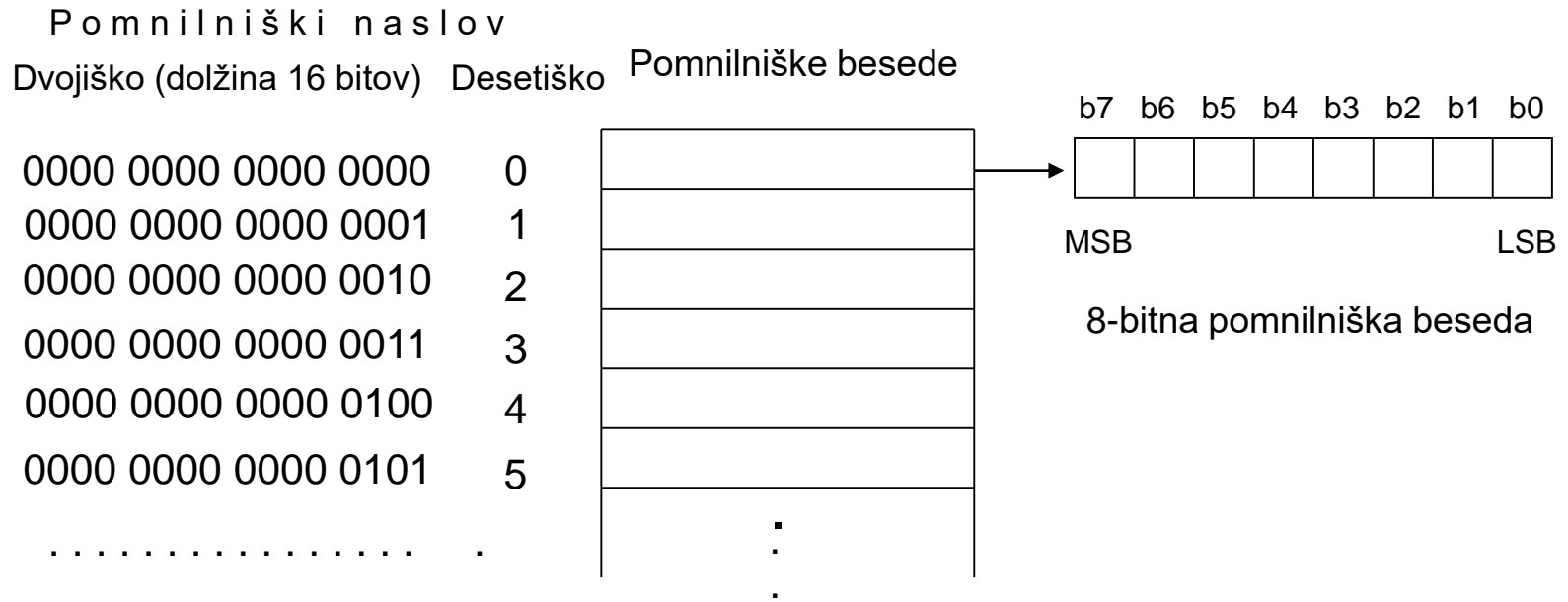
LAB 1.3 Pomnilnik

# Kaj je pomnilnik ?



Pomnilniške celice (biti)





### Pomnilniški naslov

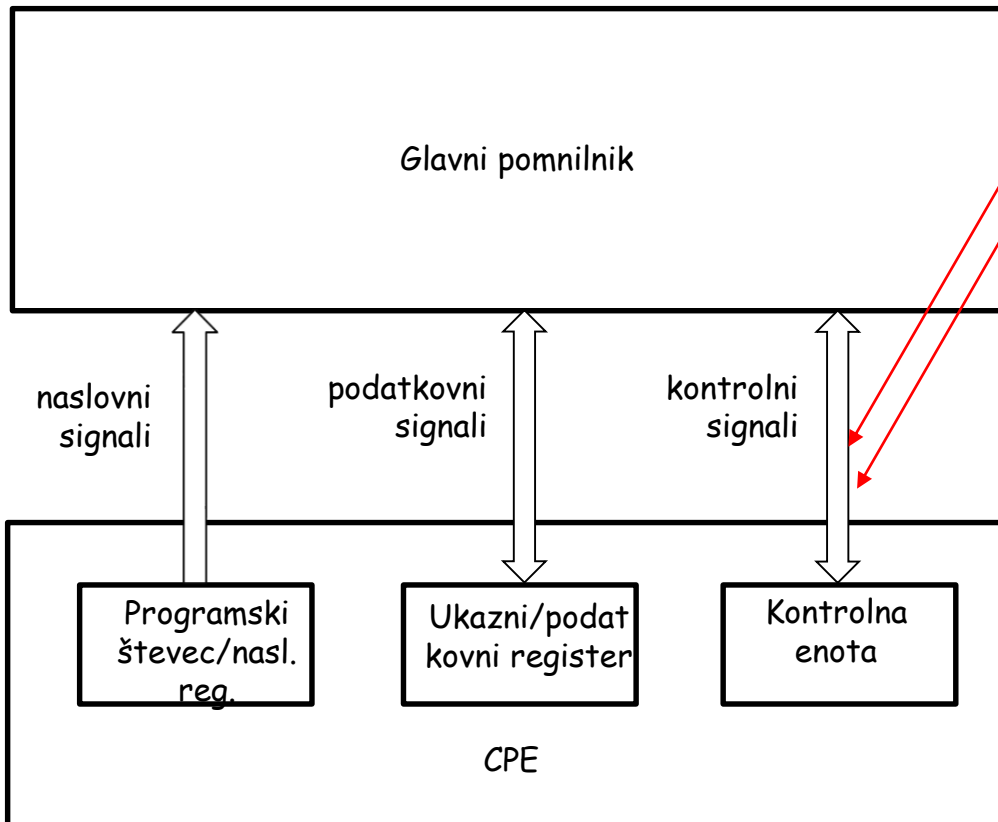
Dvojiško (dolžina 16 bitov)	Šestnajst.	Desetiško	Pomnilniške besede
0000 0000 0000 0000	0000	0	
0000 0000 0000 0001	0001	1	
0000 0000 0000 0010	0002	2	
0000 0000 0000 0011	0003	3	
0000 0000 0000 0100	0004	4	
0000 0000 0000 0101	0005	5	
.....	.		▪ : ▪
.....	.		▪ : ▪
1111 1111 1111 1011	FFFB	65531	
1111 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

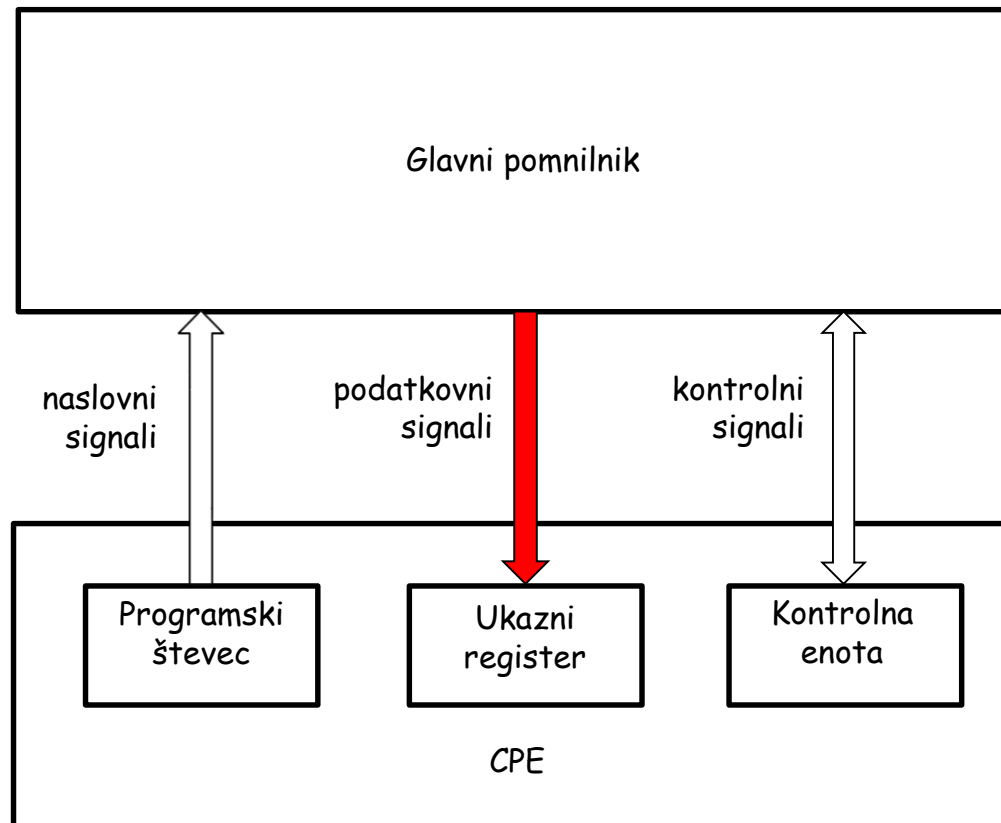
Vodilo = skupina povezav  
(naslovno, podatkovno,  
kontrolno, ...)

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



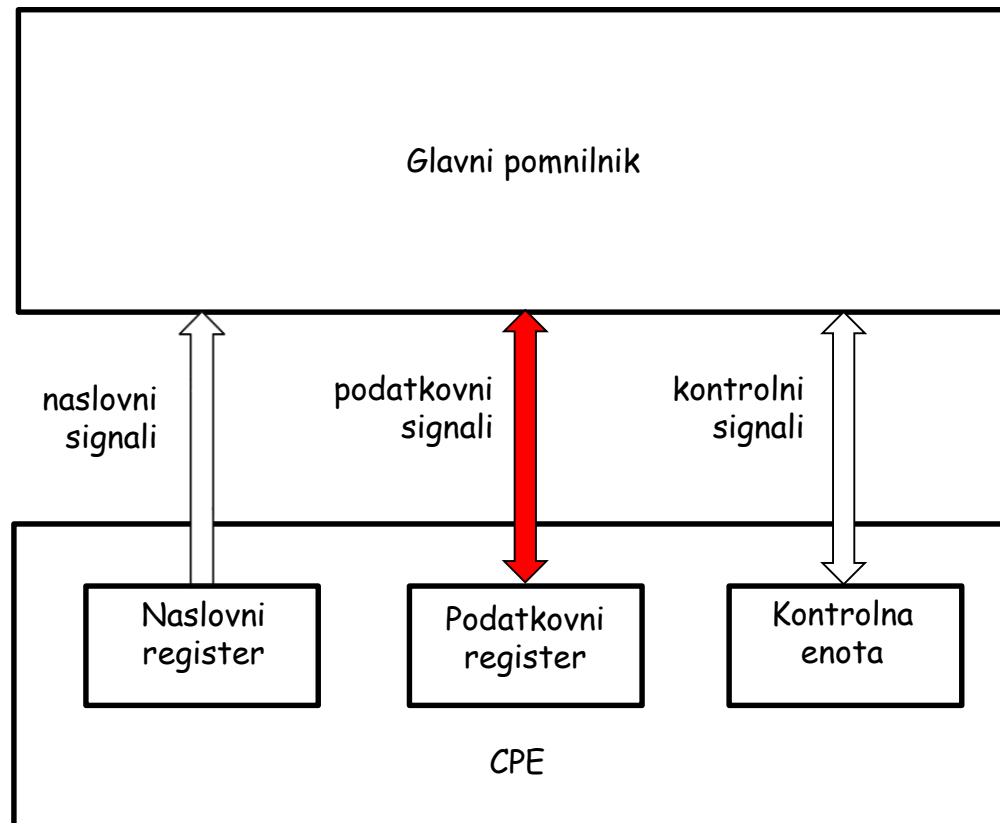
# Kako CPE dostopa do glavnega pomnilnika?

Primer za ukaze:

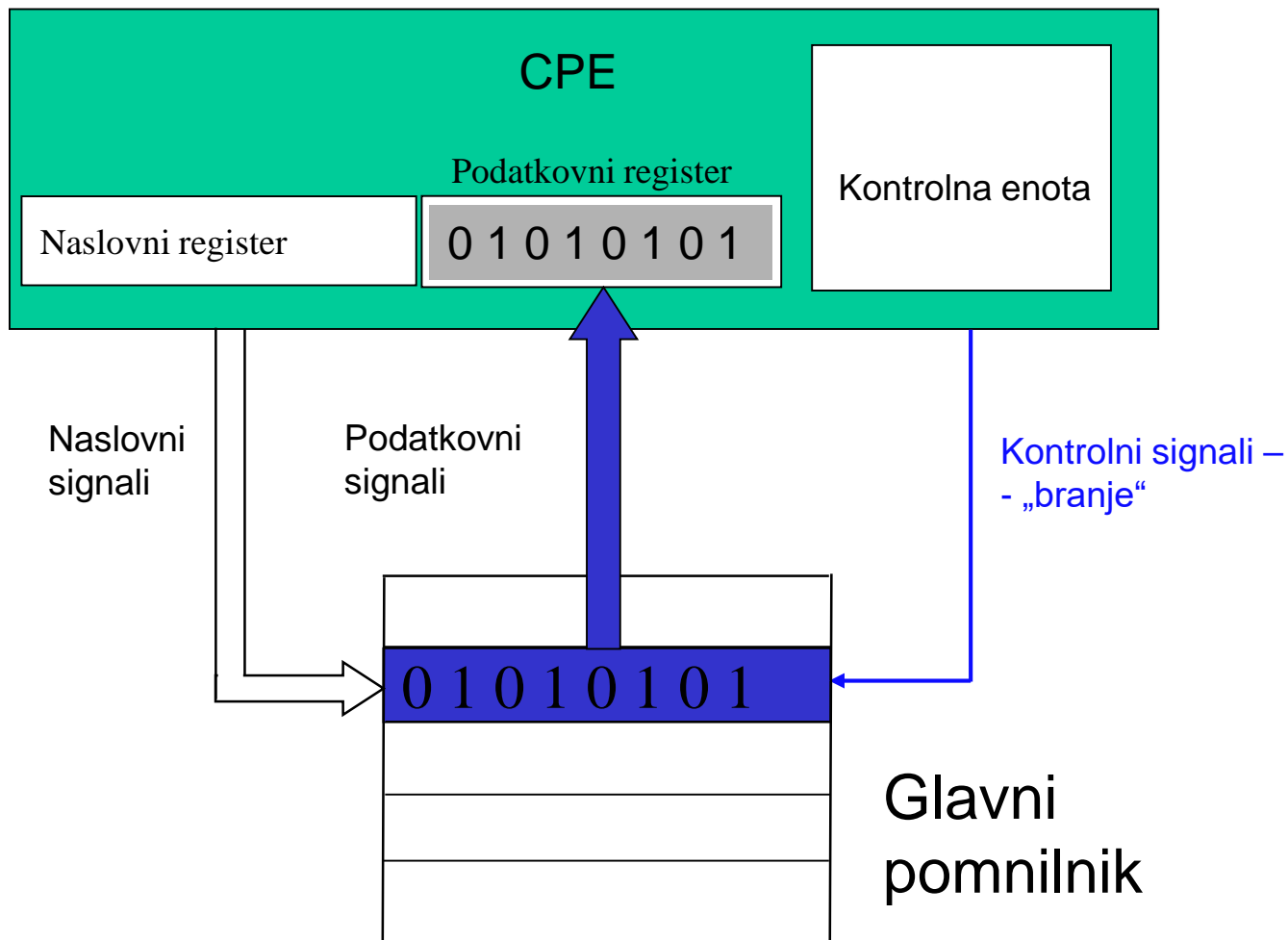


# Kako CPE dostopa do glavnega pomnilnika?

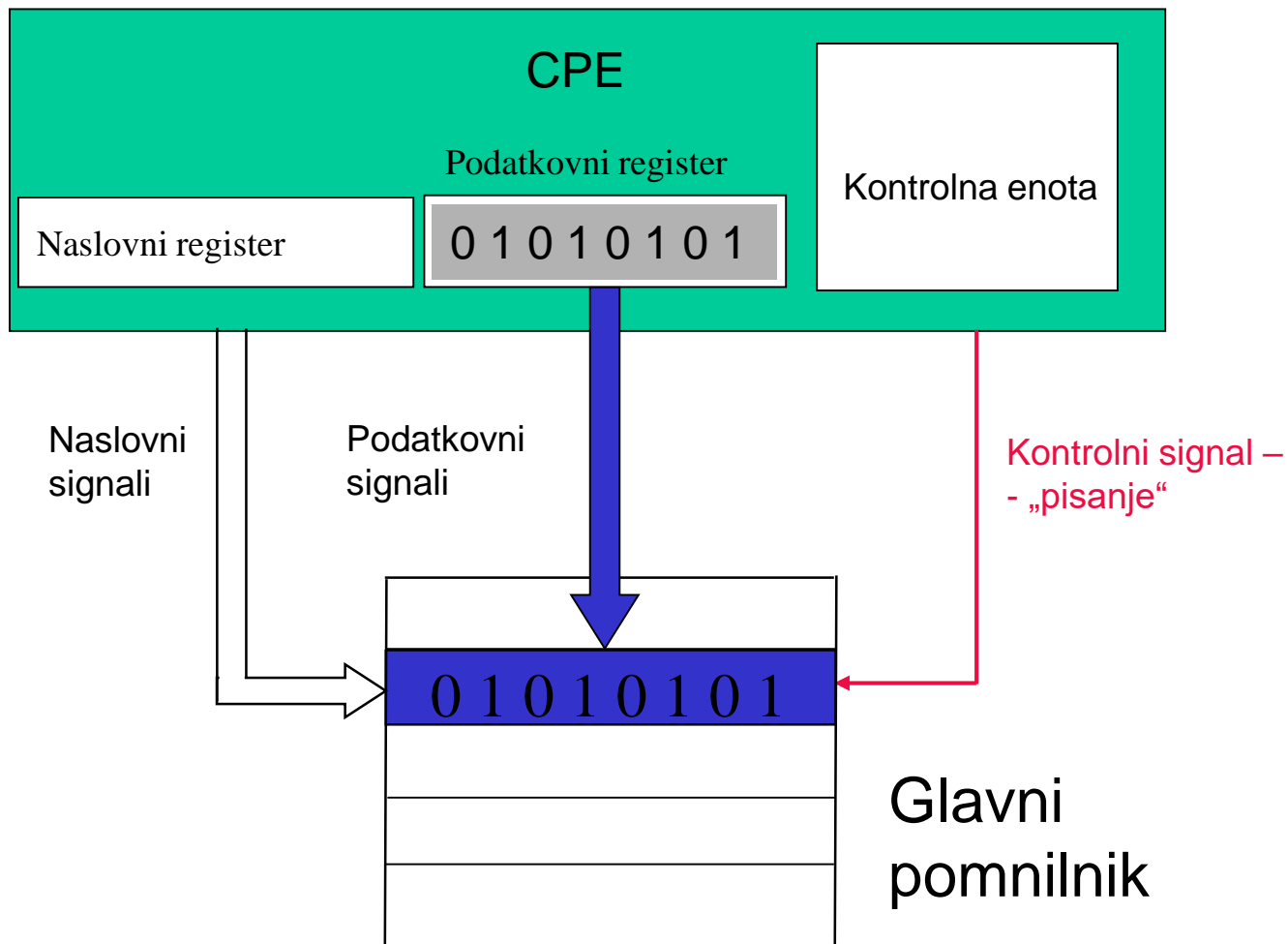
Primeri za operande:



# 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
  - Mikrokontroler AT91SAM9260 iz družine mikrokontrolerov ARM9



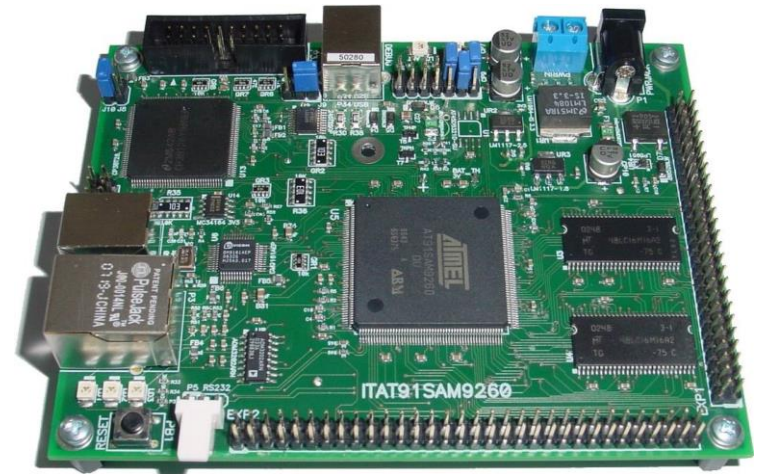
LAB 1.4 Številski sistemi (BIN,HEX) na hitro

# Računalniška arhitektura RA

Računalnik STM32H750-DK



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



LAB 1.5 Pravilo tankega/debelega konca na hitro

# Računalniška arhitektura RA

Računalnik STM32H750-DK



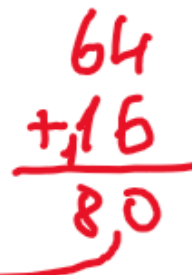
- Računalnik FRI-SMS
  - Mikrokontroler AT91SAM9260 iz družine mikrokontrolerov ARM9



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
```

```
-----
```

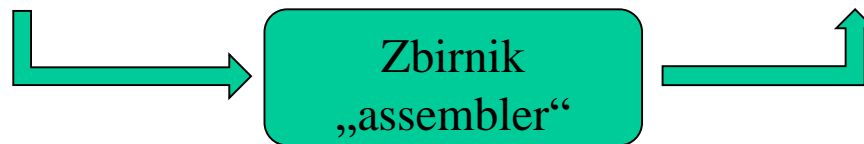
```
REZ = 0x50
```

# 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:

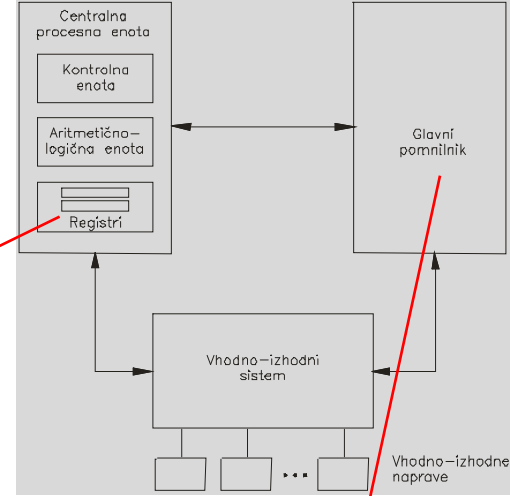
Zbirni jezik	Opis ukaza	Strojni jezik
adr r0, stev1	$R0 \leftarrow \text{nasl. stev1}$	0xE24F0014
ldr r1, [r0]	$R1 \leftarrow M[R0]$	0xE5901000
adr r0, stev2	$R0 \leftarrow \text{nasl. stev2}$	0xE24F0018
ldr r2, [r0]	$R2 \leftarrow M[R0]$	0xE5902000
add r3, r2, r1	$R3 \leftarrow R1 + R2$	0xE0823001
adr 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: Zgled

## cpulator



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**

Register	Value
r0	00000008
r1	00000040
r2	00000010
r3	00000050
r4	00000000
r5	00000000
r6	00000000
r7	00000000
r8	00000000
r9	00000000
r10	00000000
r11	00000000
r12	00000000

**Editor (Ctrl-E)**

Compile and Load (F5) Language: ARMv7 untitled.s [changed since save]

```
1 stev1: .word 0x40
2 stev2: .word 0x10
3 rez: .space 4
4
5 .global _start
6 _start:
7
8 adr r0, stev1
9 ldr r1, [r0]
10
11 adr r0, stev2
12 ldr r2, [r0]
13
14 add r3, r2, r1
15
16 adr r0, rez
17 str r3, [r0]
18
19 loop: b loop
20
21
```

**Memory (Ctrl-M)**

Go to address, label, or register:

Address	Memory contents and ASCII
00000000	aa aa aa aa aa aa aa aa
00000004	aa aa aa aa aa aa aa aa
00000008	aa aa aa aa aa aa aa aa
0000000c	aa aa aa aa aa aa aa aa
00000010	aa aa aa aa aa aa aa aa
00000014	aa aa aa aa aa aa aa aa
00000018	aa aa aa aa aa aa aa aa
0000001c	aa aa aa aa aa aa aa aa
00000020	aa aa aa aa aa aa aa aa
00000024	aa aa aa aa aa aa aa aa
00000028	aa aa aa aa aa aa aa aa
0000002c	aa aa aa aa aa aa aa aa
00000030	aa aa aa aa aa aa aa aa
00000034	aa aa aa aa aa aa aa aa
00000038	aa aa aa aa aa aa aa aa
0000003c	aa aa aa aa aa aa aa aa
00000040	aa aa aa aa aa aa aa aa
00000044	aa aa aa aa aa aa aa aa
00000048	aa aa aa aa aa aa aa aa
0000004c	aa aa aa aa aa aa aa aa
00000050	aa aa aa aa aa aa aa aa
00000054	aa aa aa aa aa aa aa aa
00000058	aa aa aa aa aa aa aa aa
0000005c	aa aa aa aa aa aa aa aa
00000060	aa aa aa aa aa aa aa aa
00000064	aa aa aa aa aa aa aa aa
00000068	aa aa aa aa aa aa aa aa
0000006c	aa aa aa aa aa aa aa aa
00000070	aa aa aa aa aa aa aa aa
00000074	aa aa aa aa aa aa aa aa
00000078	aa aa aa aa aa aa aa aa
0000007c	aa aa aa aa aa aa aa aa
00000080	aa aa aa aa aa aa aa aa
00000084	aa aa aa aa aa aa aa aa
00000088	aa aa aa aa aa aa aa aa
0000008c	aa aa aa aa aa aa aa aa
00000090	aa aa aa aa aa aa aa aa
00000094	aa aa aa aa aa aa aa aa
00000098	aa aa aa aa aa aa aa aa
0000009c	aa aa aa aa aa aa aa aa
000000a0	aa aa aa aa aa aa aa aa
000000a4	aa aa aa aa aa aa aa aa
000000a8	aa aa aa aa aa aa aa aa
000000ac	aa aa aa aa aa aa aa aa
000000b0	aa aa aa aa aa aa aa aa
000000b4	aa aa aa aa aa aa aa aa
000000b8	aa aa aa aa aa aa aa aa
000000bc	aa aa aa aa aa aa aa aa
000000c0	aa aa aa aa aa aa aa aa
000000c4	aa aa aa aa aa aa aa aa
000000c8	aa aa aa aa aa aa aa aa
000000cc	aa aa aa aa aa aa aa aa
000000d0	aa aa aa aa aa aa aa aa
000000d4	aa aa aa aa aa aa aa aa
000000d8	aa aa aa aa aa aa aa aa
000000dc	aa aa aa aa aa aa aa aa
000000e0	aa aa aa aa aa aa aa aa
000000e4	aa aa aa aa aa aa aa aa
000000e8	aa aa aa aa aa aa aa aa
000000ec	aa aa aa aa aa aa aa aa
000000f0	aa aa aa aa aa aa aa aa
000000f4	aa aa aa aa aa aa aa aa
000000f8	aa aa aa aa aa aa aa aa
000000fc	aa aa aa aa aa aa aa aa
00000100	aa aa aa aa aa aa aa aa
00000104	aa aa aa aa aa aa aa aa
00000108	aa aa aa aa aa aa aa aa
0000010c	aa aa aa aa aa aa aa aa
00000110	aa aa aa aa aa aa aa aa
00000114	aa aa aa aa aa aa aa aa
00000118	aa aa aa aa aa aa aa aa
0000011c	aa aa aa aa aa aa aa aa
00000120	aa aa aa aa aa aa aa aa
00000124	aa aa aa aa aa aa aa aa
00000128	aa aa aa aa aa aa aa aa
0000012c	aa aa aa aa aa aa aa aa
00000130	aa aa aa aa aa aa aa aa
00000134	aa aa aa aa aa aa aa aa
00000138	aa aa aa aa aa aa aa aa
0000013c	aa aa aa aa aa aa aa aa
00000140	aa aa aa aa aa aa aa aa
00000144	aa aa aa aa aa aa aa aa
00000148	aa aa aa aa aa aa aa aa
0000014c	aa aa aa aa aa aa aa aa
00000150	aa aa aa aa aa aa aa aa
00000154	aa aa aa aa aa aa aa aa
00000158	aa aa aa aa aa aa aa aa
0000015c	aa aa aa aa aa aa aa aa
00000160	aa aa aa aa aa aa aa aa
00000164	aa aa aa aa aa aa aa aa
00000168	aa aa aa aa aa aa aa aa
0000016c	aa aa aa aa aa aa aa aa
00000170	aa aa aa aa aa aa aa aa
00000174	aa aa aa aa aa aa aa aa
00000178	aa aa aa aa aa aa aa aa
0000017c	aa aa aa aa aa aa aa aa
00000180	aa aa aa aa aa aa aa aa
00000184	aa aa aa aa aa aa aa aa
00000188	aa aa aa aa aa aa aa aa
0000018c	aa aa aa aa aa aa aa aa
00000190	aa aa aa aa aa aa aa aa

**Messages**

Code and data loaded from ELF-executable into memory. Total size is 48 bytes.

Assemble: arm-altera-eabi-as -mfloat-abi=soft -march=armv7-a -mcpu=cortex-a9 -mfpu=neon-fp16 --gdwarf2 -o work/asmV0FoCU.s.o work/a  
Link: arm-altera-eabi-ld --script build\_arm.ld -e \_start -u \_start -o work/asmV0FoCU.s.elf work/asmV0FoCU.s.o  
Compile succeeded.



# 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
0x00 = 0		STEV1
0x04 = 4		STEV2
0x08 = 8		REZ
0x0C = 12		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

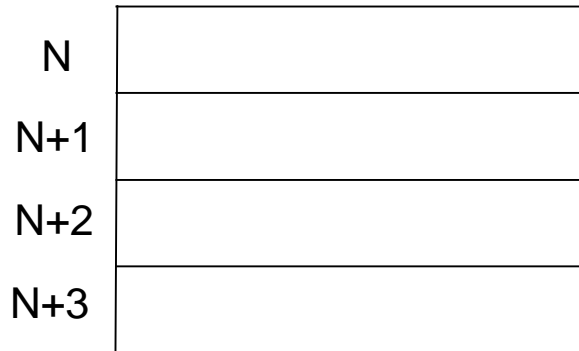
---

MSB

LSB

0 x AA BB CC DD

Debeli konec  
Big Endian



Tanki konec  
Little Endian

