



RAČUNALNIŠKA ARHITEKTURA

3 Osnove delovanja računalnikov



3 Osnove delovanja računalnikov - vsebina

- Von Neumannov računalniški model
 - Von Neumannov računalniški model
 - Delovanje von Neumannovega računalnika
- Flynnova klasifikacija
- Glavni pomnilnik v von Neumannovem računalniku
 - Pomnilniška beseda
 - Pomnilniški naslov
 - Naslovni prostor
 - Vsebina pomnilniške besede
 - Princetonska in harvardska pomnilniška arhitektura
 - Dostop do pomnilnika
- Amdahlov zakon
- Jeziki, nivoji in navidezni računalniki
 - Računalnik kot zaporedje navideznih računalnikov
 - Prehajanje iz jezika J2 v jezik J1
 - Strojna in programska oprema računalnika
- Primer izvedbe programa v računalniku



Osnove delovanja računalnikov - vsebina:

- ☐ 3.1 Von Neumannov računalniški model
- ☐ 3.2 Flynnova klasifikacija
- ☐ 3.3 Glavni pomnilnik v von Neumannovem računalniku
- ☐ 3.4 Amdahlov zakon
- ☐ 3.5 Jeziki, nivoji in navidezni računalniki
- ☐ 3.6 Primer izvedbe programa v računalniku



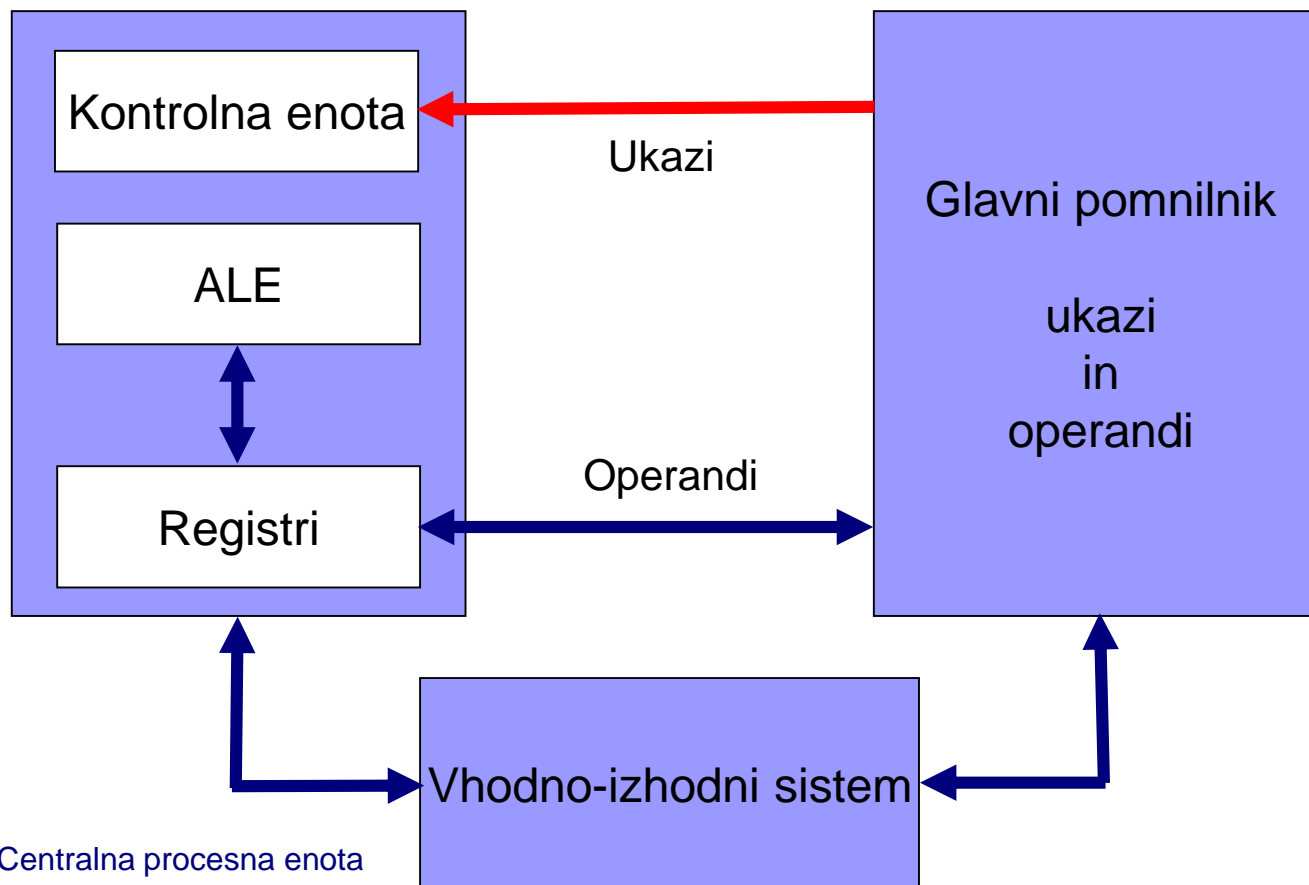
3.1 Von Neumannov računalniški model

- Sestavljajo ga trije osnovni deli:
 - CPE (centralna procesna enota)
 - Glavni pomnilnik
 - Vhodno-izhodni (V/I) sistem
- Je stroj s **shranjenim programom**, ki je shranjen v glavnem pomnilniku. Ukazi v programu določajo, kaj bo stroj delal.
- **Program vodi delovanje stroja** – program določa, kako bo stroj deloval.
- CPE jemlje ukaze iz glavnega pomnilnika in jih izvaja drugega za drugim.



Von Neumannov računalniški model

CPE



CPE – Centralna procesna enota
ALE – Aritmetično logična enota



- **CPE** iz glavnega pomnilnika jemlje ukaze in jih izvršuje. V današnjih računalnikih je poleg CPE še več procesorjev, zato oznaka **centralna** procesna enota. Sestavljajo jo trije deli:
 - KONTROLNA ENOTA - skrbi za prevzemanje ukazov in operandov in aktiviranje operacij, ki so določene z ukazi.
 - ALE - izvaja aritmetične operacije (seštevanje . . .)
in logične operacije (AND . . .).
 - REGISTRI – več povezanih pomnilniških celic, ki služijo za shranjevanje vrednosti.
 - Programsko nedostopni registri – potrebni za delovanje CPE.
 - Programsko dostopni registri (arhitekturni registri) za shranjevanje operandov. Predstavljajo majhen in hiter pomnilnik v CPE.



- **Glavni pomnilnik** je sestavljen iz pomnilniških besed. Vsaka pomnilniška beseda ima svoj enolični naslov.
 - V njem so shranjeni ukazi in operandi.
 - Oznaka glavni zopet služi za razlikovanje od drugih pomnilnikov v današnjih računalnikih (predpomnilniki, navidezni pomnilnik).

- **V/I sistem** za prenos informacije v zunanji svet ali iz zunanjega sveta. Informacija je v CPE in glavnem pomnilniku shranjena v obliki, ki ni dostopna zunanjemu svetu.
 - Sestavni del V/I sistema so vhodno-izhodne naprave, ki pretvarjajo informacijo v neko drugo obliko, ki je primerna za uporabnika ali pa služijo kot pomožni pomnilniki.



Delovanje von Neumannovega računalnika

- Njegovo delovanje popolnoma določajo ukazi (strojni ukazi), ki jih CPE jemlje iz glavnega pomnilnika zaporedoma enega za drugim.
- Strojni ukazi so v pomnilniku shranjeni eden za drugim po naraščajočih naslovih.
- Na neki način je določeno, iz katerega naslova se vzame prvi ukaz po vklopu računalnika ali po pritisku na tipko RESET.
 - Najenostavneje: prva ali zadnja pomnilniška lokacija – najnižji ali najvišji naslov v pomnilniku.

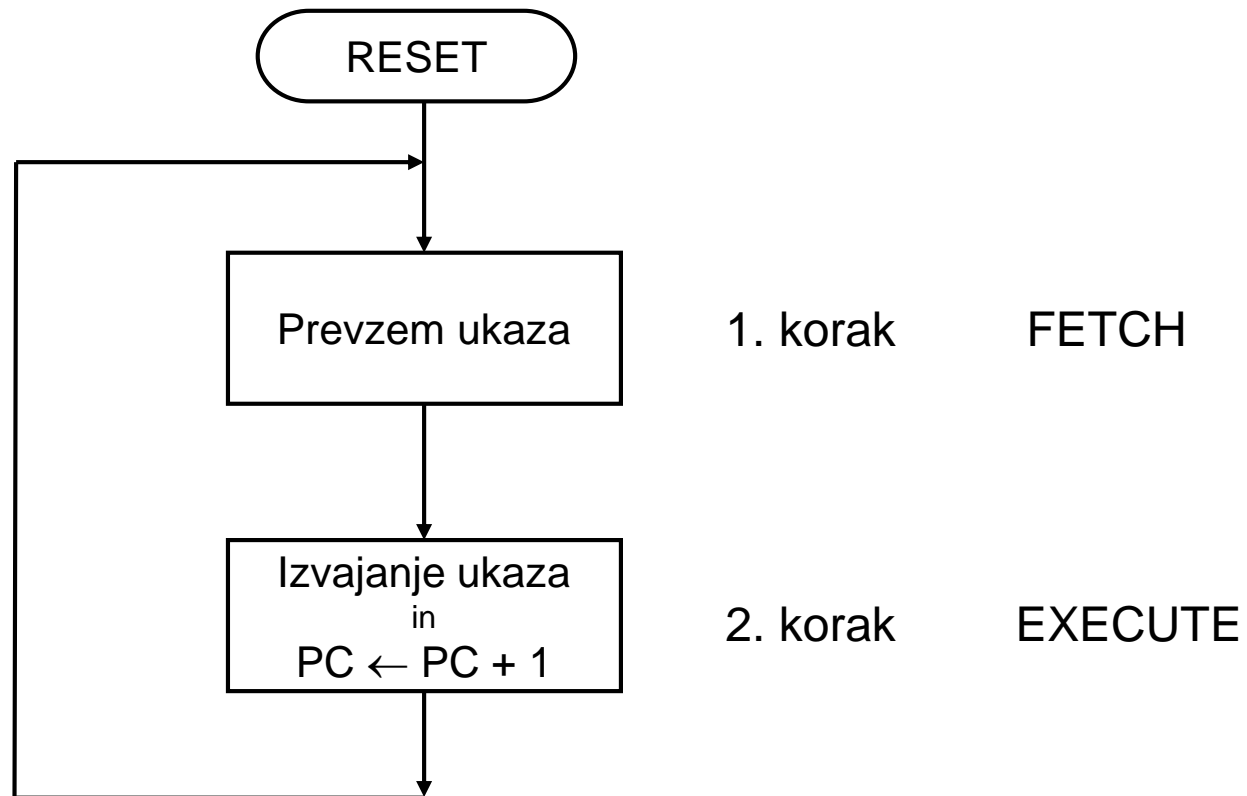


Pri vsakem ukazu razlikujemo dva koraka

- **1. korak:** Jemanje ukaza iz pomnilnika (FETCH)
(tudi branje ali prevzem ukaza)
 - ukazno prevzemni cikel
 - angl. fetch cycle
- V CPE je poseben register - programski števec (PC - Program Counter), ki vedno vsebuje pomnilniški naslov, na katerem je v pomnilniku shranjen naslednji ukaz.
- **2. korak:** Izvrševanje v 1. koraku prevzetega ukaza (EXECUTE)
 - izvršilni cikel
 - angl. execute cycle



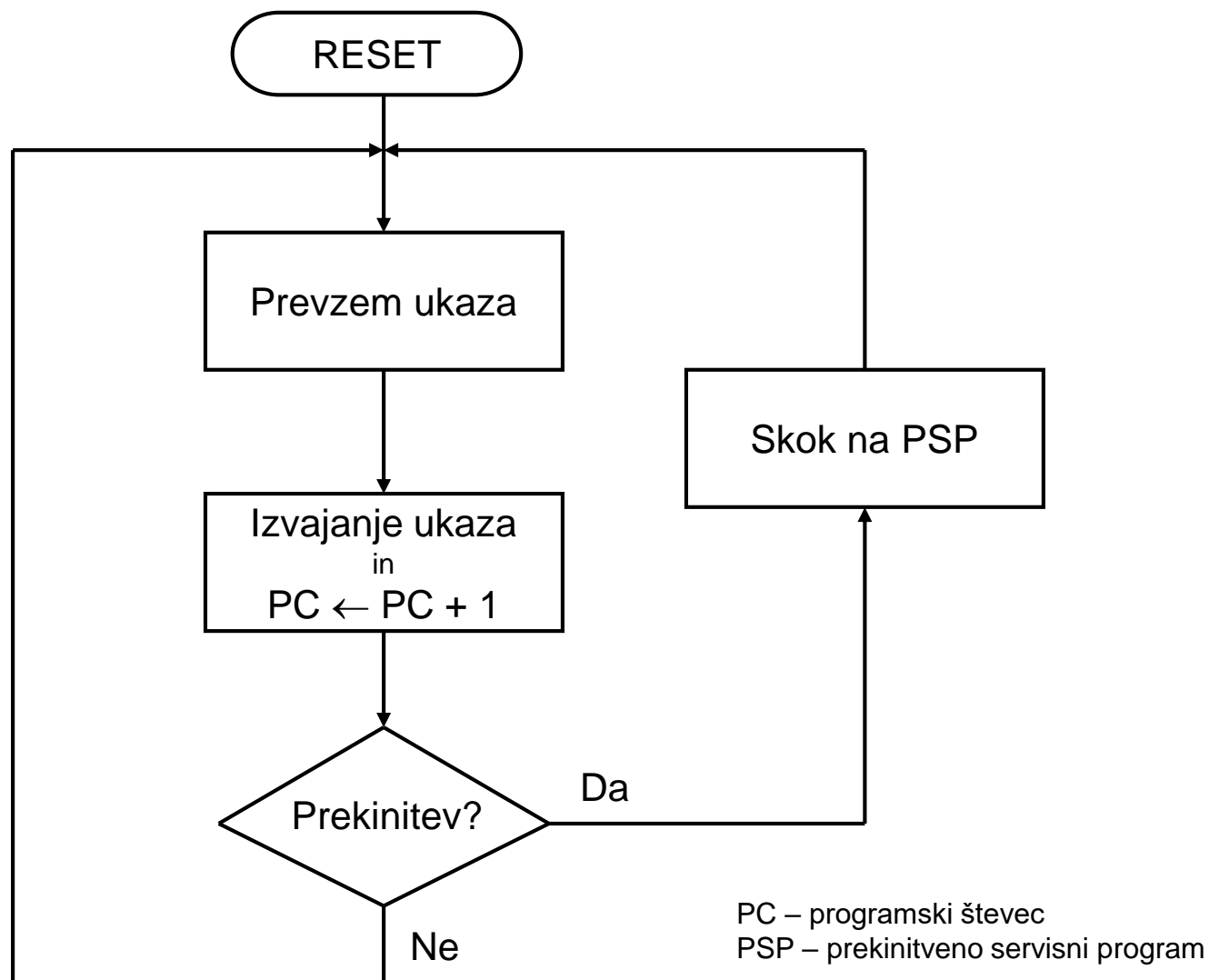
- Vsak ukaz vsebuje dve vrsti informacij:
 - ☐ informacijo o operaciji, ki naj se izvrši,
 - ☐ informacije o operandih, nad katerimi naj se operacija izvrši.
- CPE izvrši operacijo in poskrbi, da je v PC naslov naslednjega ukaza tako, da poveča vsebino PC-ja za 1 (ali več).
- **Pravilo:** ukazi v pomnilniku so shranjeni po naraščajočih naslovih zato $PC \leftarrow PC + 1$. To pravilo je rezultat dogovora in določa vrstni red izvajanja ukazov.





Po zaključku koraka 2 začne CPE zopet s korakom 1.
Ta dva koraka se ponavljata, dokler računalnik deluje.

- **Izjema 1:** Skočni ukazi, s katerimi lahko v PC zapišemo poljuben naslov.
- **Izjema 2:** Prekinitev ali past
CPE po koraku 2 ne prevzame ukaza po pravilu $PC \leftarrow PC + 1$, temveč začne izvajati drug program - prekinitveno servisni program (PSP).





- Zaporedno izvajanje ukazov je počasno in predstavlja osnovno slabost von Neumannovih računalnikov.
- Razširitve osnovnega von Neumannovega modela so zajete v Flynnovi klasifikaciji.



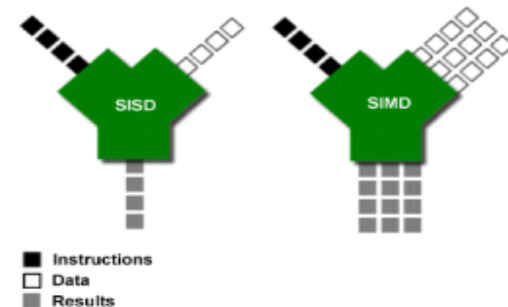
Osnove delovanja računalnikov - vsebina:

- ☐ 3.1 Von Neumannov računalniški model
- ☐ 3.2 Flynnova klasifikacija
- ☐ 3.3 Glavni pomnilnik v von Neumannovem računalniku
- ☐ 3.4 Amdahlov zakon
- ☐ 3.5 Jeziki, nivoji in navidezni računalniki
- ☐ 3.6 Primer izvedbe programa v računalniku



3.2 Flynnova klasifikacija

- To razvrstitev računalnikov v štiri skupine je predlagal M.J.Flynn leta 1966. Osnovna kriterija te klasifikacije za razvrstitev računalnikov sta:
 - ☐ število ukazov, ki se izvršujejo hkrati (instruction stream),
 - ☐ število operandov, ki jih en ukaz hkrati obdeluje (data stream).
- Po teh dveh kriterijih spada vsak računalnik v enega od štirih razredov:
- 1 SISD (Single Instruction Single Data)
 - ☐ klasični Von Neumannovi računalniki brez paralelizma pri ukazih in operandih
 - ☐ Intel Pentium 4



Source: ARS Technica

■ 2 SIMD (Single Instruction Multiple Data)

- ☐ Pravi vektorski računalniki (paralelni računalniki, grafični procesorji)
- ☐ Ukazi SSE (Streaming SIMD Extensions) pri procesorjih z arhitekturo x86

■ 3 MISD ! (Multiple Instruction Single Data)

- ☐ Neobičajna arhitektura. Več ukazov nad istimi operandi – uporablja se kjer se zahteva neobčutljivost na napake.

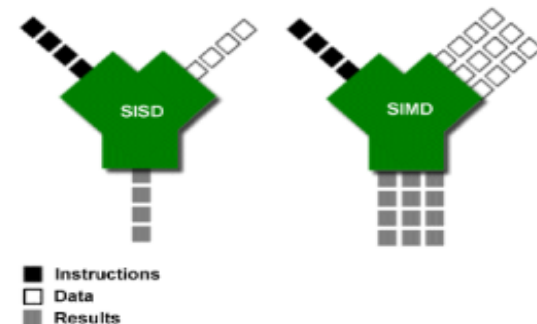
■ 4 MIMD (Multiple Instruction Multiple Data)

- ☐ Multiprocesorski računalniki (paralelni računalniki)
- ☐ V to skupino se (pogojno) uvrščajo tudi večjedrni superskalarni računalniki, npr. Intel Core i7, čeprav jih zaradi omejenega števila jeder običajno uvrstimo v SISD.





- Pri MIMD računalnikih se naenkrat izvaja več ukazov, vsak na svojih operandih.
- MIMD računalnik tvori več povezanih navadnih von Neumannovih računalnikov – več CPE, ki so med seboj povezane.
- Večjedrne računalnike včasih štejemo tudi kar med SISD, čeprav jih lahko uvrščamo tudi v SIMD in MIMD.



Source: ARS Technica

Primer :

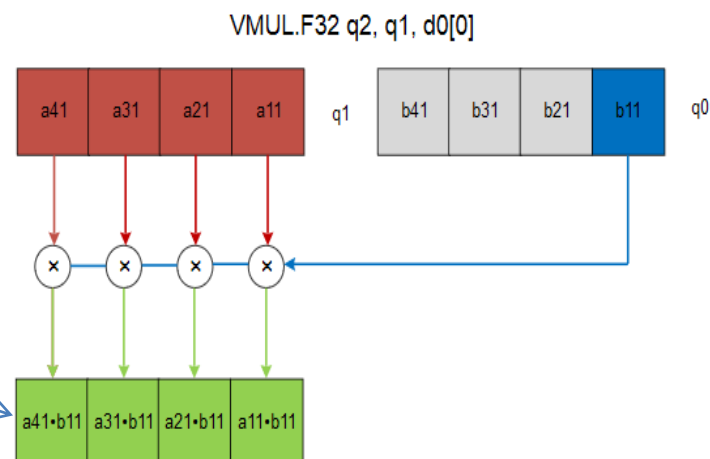
■ SIMD enota (Single Instruction Multiple Data)

Primer: matrično množenje: (ARM: NEON enota kot SIMD razširitev) :

Figure 4.4. Matrix multiplication showing one column of results

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \cdot \begin{bmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ b_{31} & b_{32} & b_{33} & b_{34} \\ b_{41} & b_{42} & b_{43} & b_{44} \end{bmatrix} = \begin{bmatrix} a_{11} \cdot b_{11} + a_{12} \cdot b_{21} + a_{13} \cdot b_{31} + a_{14} \cdot b_{41} & \dots & \dots & \dots \\ a_{21} \cdot b_{11} + a_{22} \cdot b_{21} + a_{23} \cdot b_{31} + a_{24} \cdot b_{41} & \dots & \dots & \dots \\ a_{31} \cdot b_{11} + a_{32} \cdot b_{21} + a_{33} \cdot b_{31} + a_{34} \cdot b_{41} & \dots & \dots & \dots \\ a_{41} \cdot b_{11} + a_{42} \cdot b_{21} + a_{43} \cdot b_{31} + a_{44} \cdot b_{41} & \dots & \dots & \dots \end{bmatrix}$$

Figure 4.5. NEON vector-by-scalar multiplication



GPU : podobna filozofija, je širši koncept

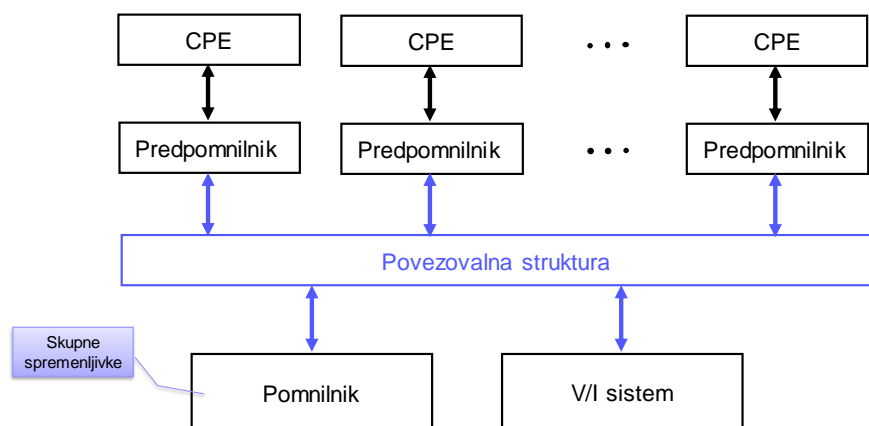




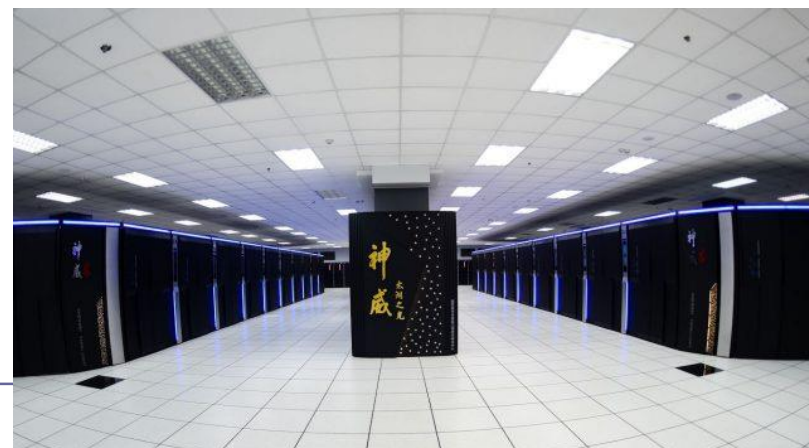
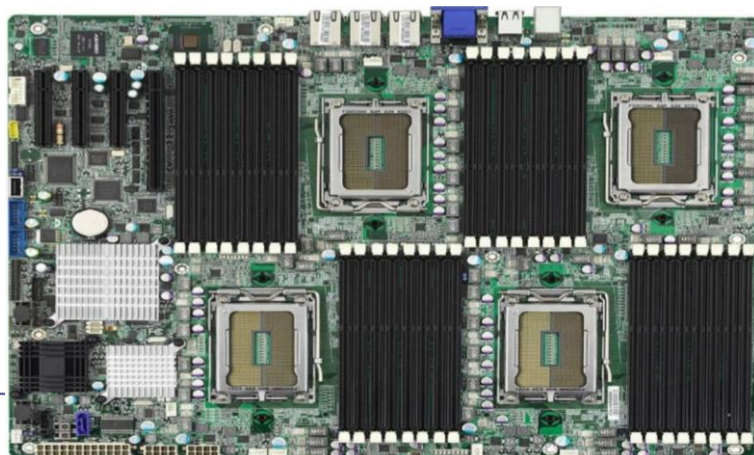
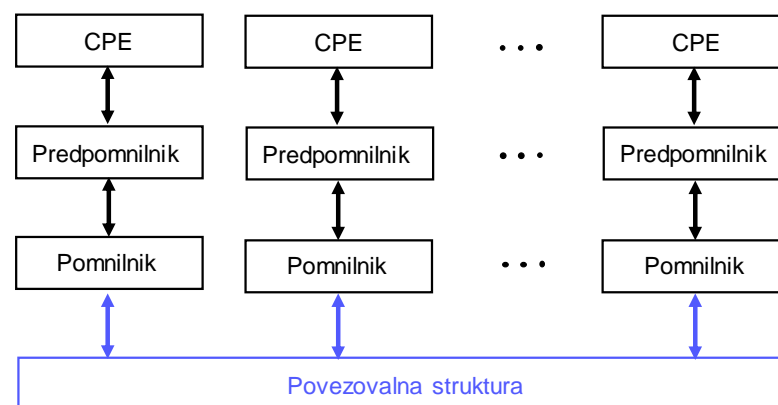
Primeri:

■ 4 MIMD (Multiple Instruction Multiple Data)

Multiprocesor (tesneje povezan)



Multiračunalnik (ohlapneje povezan)





Osnove delovanja računalnikov - vsebina:

- ☐ 3.1 Von Neumannov računalniški model
- ☐ 3.2 Flynnova klasifikacija
- ☐ 3.3 Glavni pomnilnik v von Neumannovem računalniku
- ☐ 3.4 Amdahlov zakon
- ☐ 3.5 Jeziki, nivoji in navidezni računalniki
- ☐ 3.6 Primer izvedbe programa v računalniku



3.3 Glavni pomnilnik v von Neumannovem računalniku

■ Definicija

- Glavni pomnilnik je **pasivna naprava** in služi za shranjevanje ukazov in operandov.
- Osnovna celica v pomnilniku je pomnilniška celica, ki lahko hrani 1 bit informacije (vsebina 0 ali 1).

■ Pomnilniška beseda (tudi pomnilniška lokacija)

- Pomnilniška beseda je definirana kot najmanjše število bitov, ki imajo svoj naslov. Pomnilniška beseda je torej najmanjša naslovljiva enota v pomnilniku.
- Pomnilnik je enodimenzionalno zaporedje pomnilniških besed.
- Pomnilniško besedo sestavlja določeno število enobitnih pomnilniških celic.
- **Dolžina pomnilniške besede** je število enobitnih pomnilniških celic, ki sestavljajo pomnilniško besedo. Danes je najpogostejša dolžina besede 1 bajt (= 8 bitov).



■ Pomnilniški naslov

- ☐ Je enolična oznaka vsake pomnilniške besede
- ☐ Vsaka pomnilniška beseda ima svoj enolični pomnilniški naslov.
- ☐ Naslov pomnilniške besede je nespremenljiv.
- ☐ Število bitov, ki sestavljajo naslov, imenujemo **dolžina naslova**.
- ☐ Dolžina naslova v bitih določa naslovni prostor.

■ Naslovni prostor (tudi pomnilniški prostor)

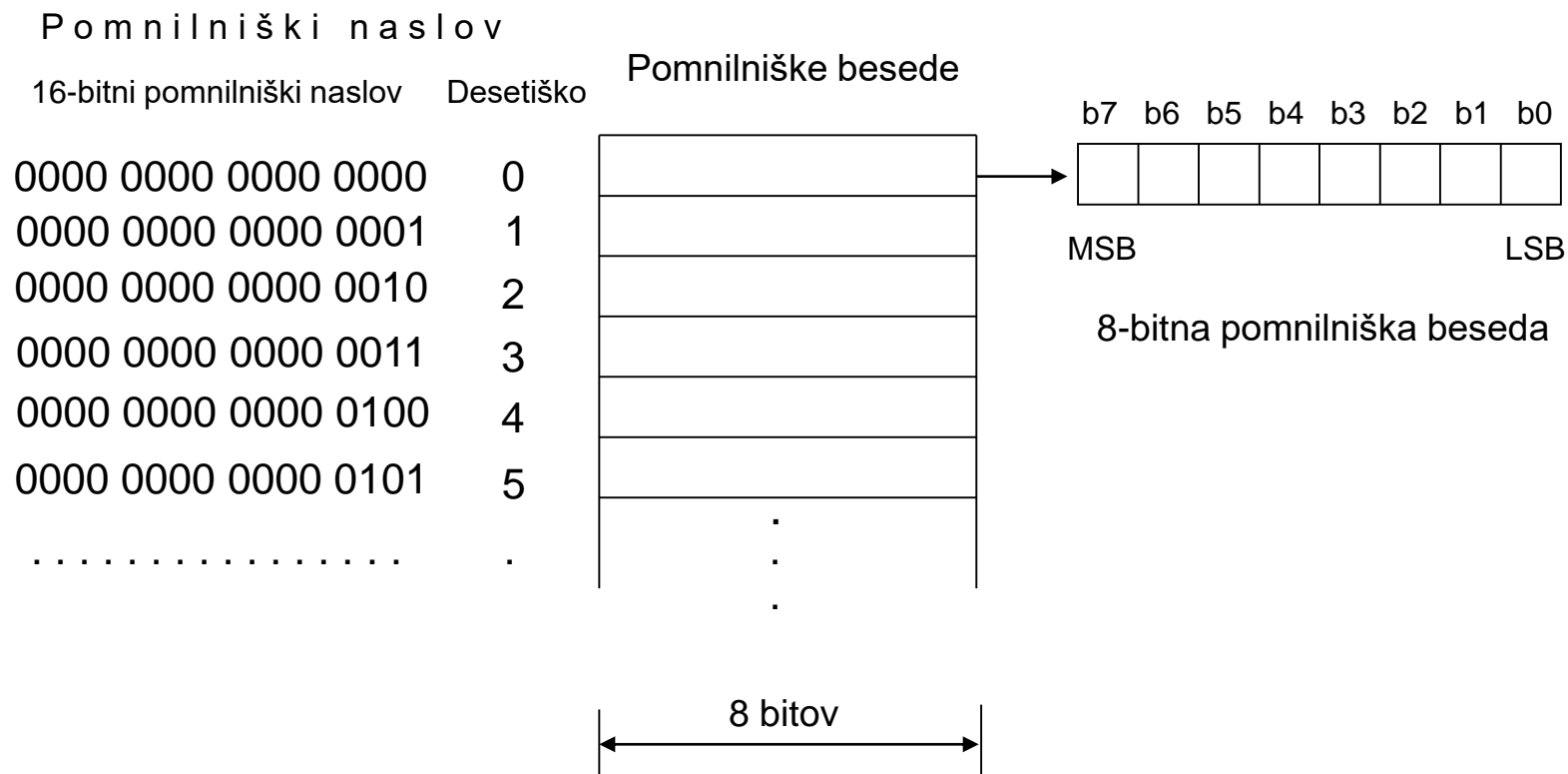
- ☐ Je množica vseh naslovov
- ☐ In določa tudi največjo možno velikost pomnilnika.



- **Vsebina pomnilniške besede** se lahko spreminja. V 8-bitno pomnilniško besedo lahko shranimo $2^8 = 256$ različnih vsebin.
- Naslov pomnilniške besede pa je nespremenljiv.
- Število pomnilniških besed v glavnem pomnilniku ni nujno enako velikosti naslovnega prostora.
- Deli naslovnega prostora so lahko prazni (vsi naslovi niso uporabljeni) \Rightarrow glavni pomnilnik je običajno manjši od največje možne velikosti.



Glavni pomnilnik v von Neumannovem računalniku





Pomnilniški naslov

Dvojiško (16-bitni naslov)	Šestnajstiško	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	

64 K (= 2¹⁶) pomnilniških besed



Predpone kilo, mega, giga idr. so samo pri velikosti pomnilnika potence števila 2!

Pomnilniki

- 1K (kilo) = $2^{10} = 1024$ (1 KB = 1024 B)
- 1M (mega) = $2^{20} = 1\,048\,576$ (1 MB = 1 048 576 B)
- 1G (giga) = $2^{30} = 1\,073\,741\,824$ (1 GB = $1024 \cdot 1024 \cdot 1024 = 1\,073\,741\,824$ B)
 - Vzrok je tehnološki: npr. 10-bitni pomnilniški naslov omogoča $2^{10} = 1024$ različnih naslovov in ne 1000.
 - Predlog IEC 1998: KiB = 2^{10} B, MiB = 2^{20} B, GiB = 2^{30} B

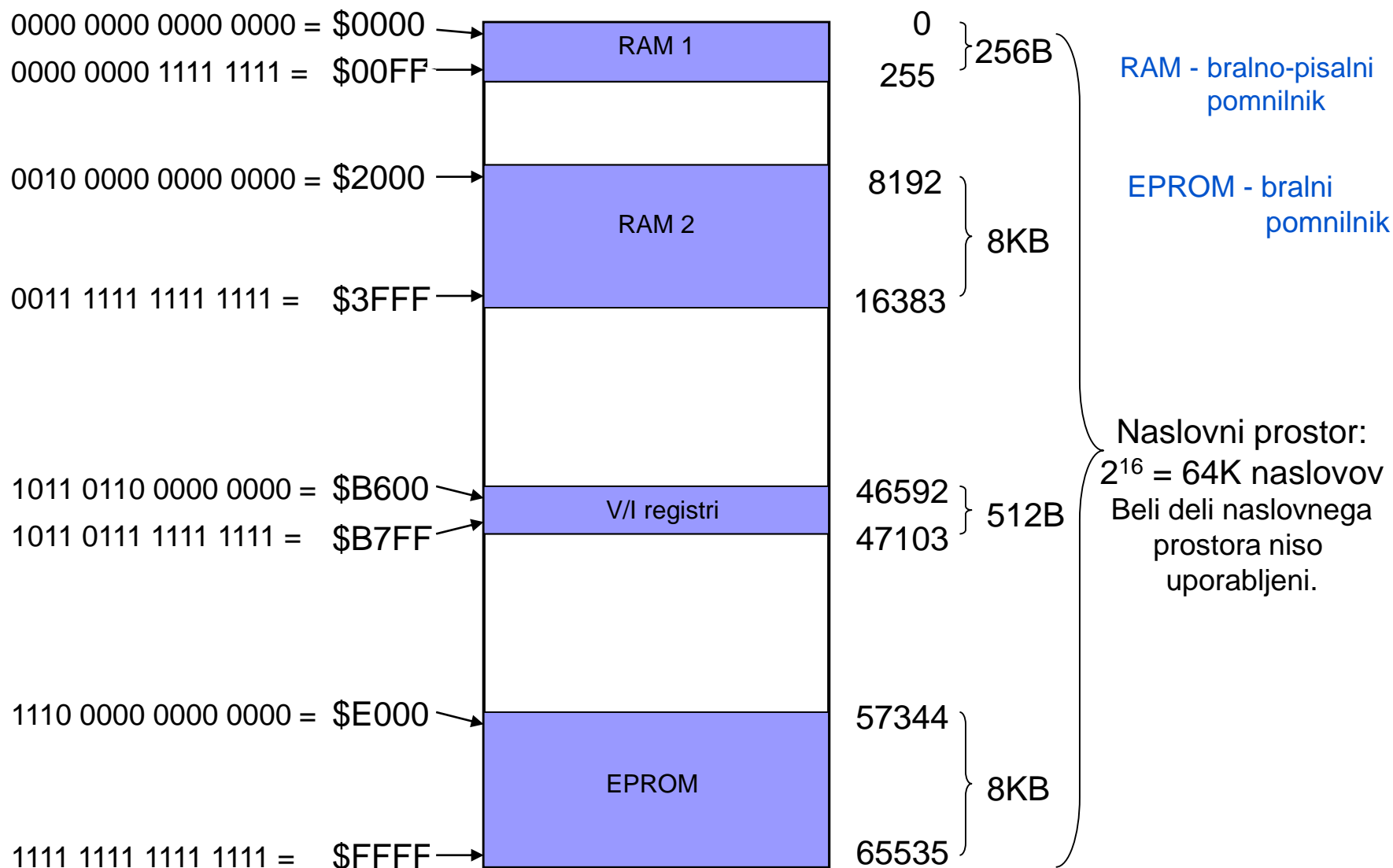
Druga področja (frekvenca, hitrost prenosa ...)

- 1k (kilo) = $10^3 = 1000$ (1 km = 1000 m)
- 1M (mega) = $10^6 = 1\,000\,000$ (100 Mb/s = 100 000 000 b/s)
- 1G (giga) = $10^9 = 1\,000\,000\,000$ (1 GHz = 1 000 000 000 Hz)



Primer slike pomnilnika (memory map) pri procesorju 68HC11 – procesor ima 16-bitni pomnilniški naslov

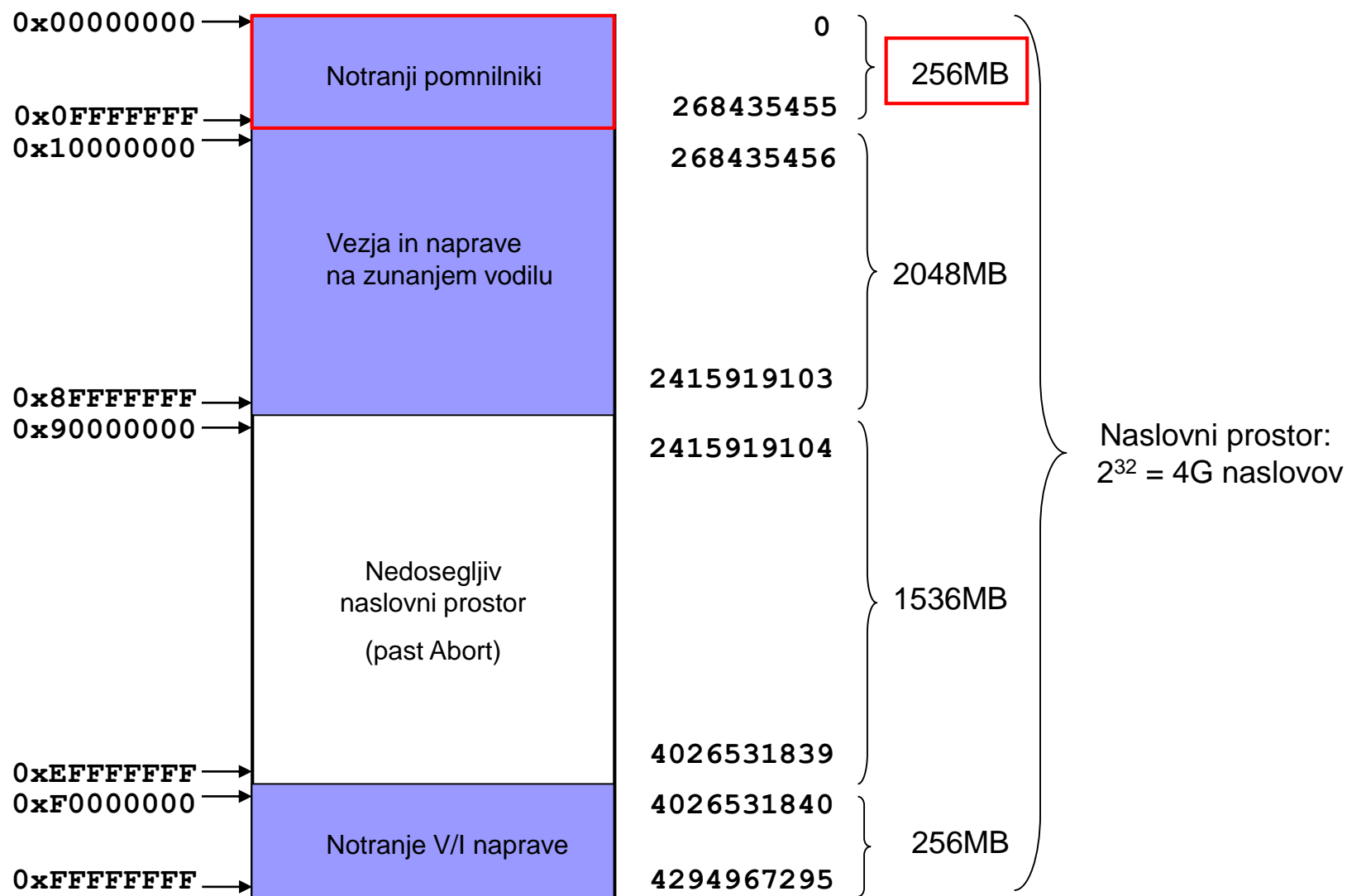
16-bitni pomnilniški naslov





Primer slike pomnilnika pri procesorju AT91SAM9260 (32-bitni pomnilniški naslov)

32-bitni naslov - 8 hex znakov





Slika pomnilnika pri AT91SAM9260 (memory map)

32-bitni pomnilniški naslov - 8 hex znakov

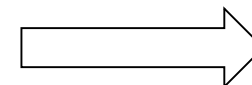
0x00000000 →

0x0FFFFFFF →

Notranji pomnilniki

0
268435455 }

256MB

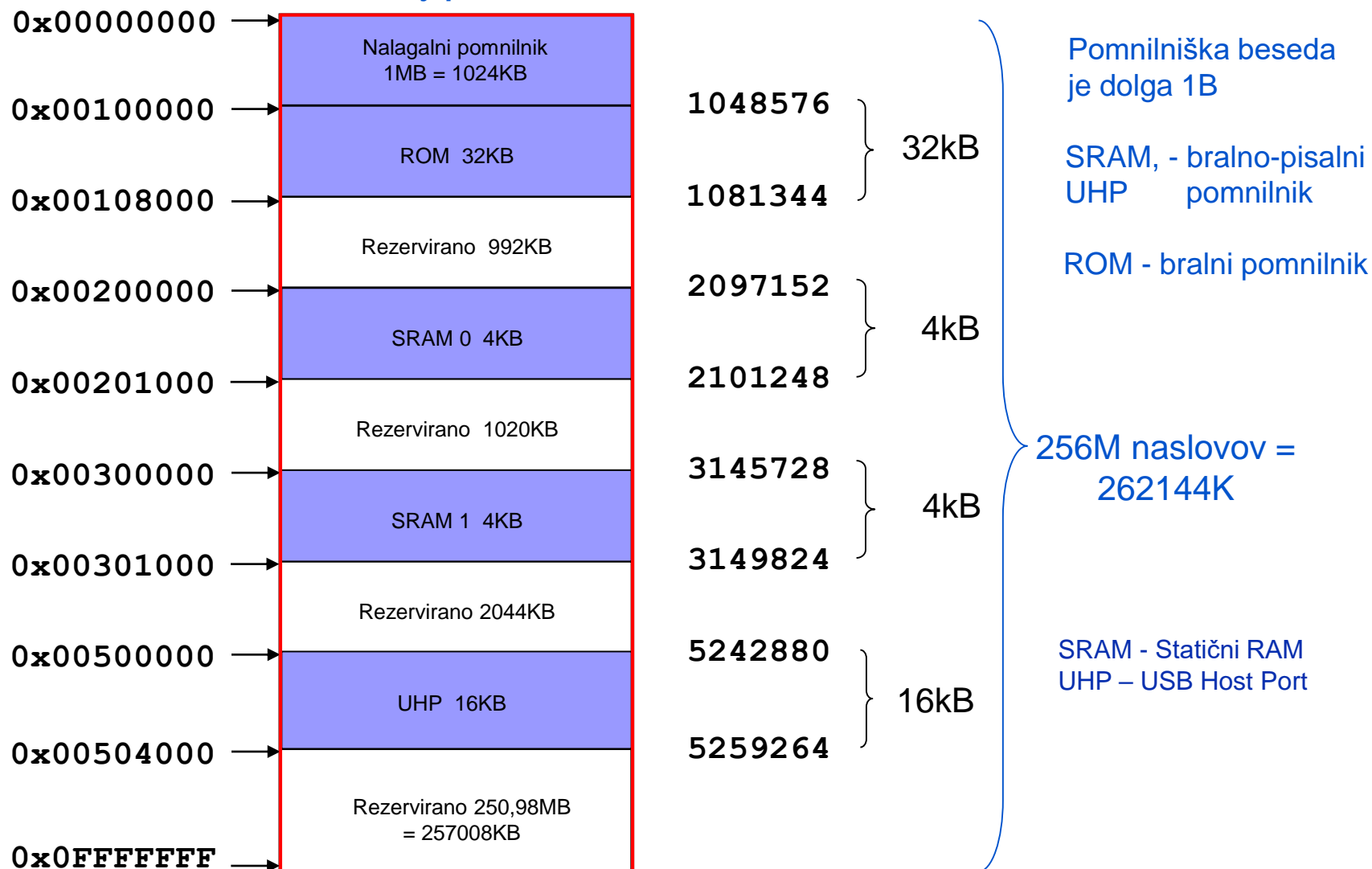


Naslovni prostor:
 $2^{32} = 4\text{G}$ naslovov



Slika notranjega pomnilnika (prvih 256 MB) do pri AT91SAM9260

Notranji pomnilniki





Von Neumannovo ozko grlo

- Prenosi CPE \leftrightarrow gl. pomnilnik - promet
- Von Neumannovo ozko grlo - povezava med CPE in glavnim pomnilnikom. Iz pomnilnika v CPE se prenašajo vsi ukazi in operandi iz pomnilnika ali v pomnilnik.
- Eden od načinov za razširitev tega ozkega grla je razdelitev glavnega pomnilnika v dva dela.



- Pomnilnik je pri harvardski arhitekturi razdeljen na dva ločena pomnilnika.
- V enem so shranjeni samo operandi – operandni pomnilnik, v drugem pa samo ukazi – ukazni pomnilnik.
- Ukazni in operandni pomnilnik lahko delujeta istočasno. Tako lahko dosežemo do dvakrat večjo hitrost.
- Harvardska arhitektura se danes uporablja pri predpomnilniku na najnižjem nivoju (operandni in ukazni predpomnilnik L1), glavni pomnilnik pa je pri večini računalnikov en sam (princetonska arhitektura).



Dostop do pomnilnika

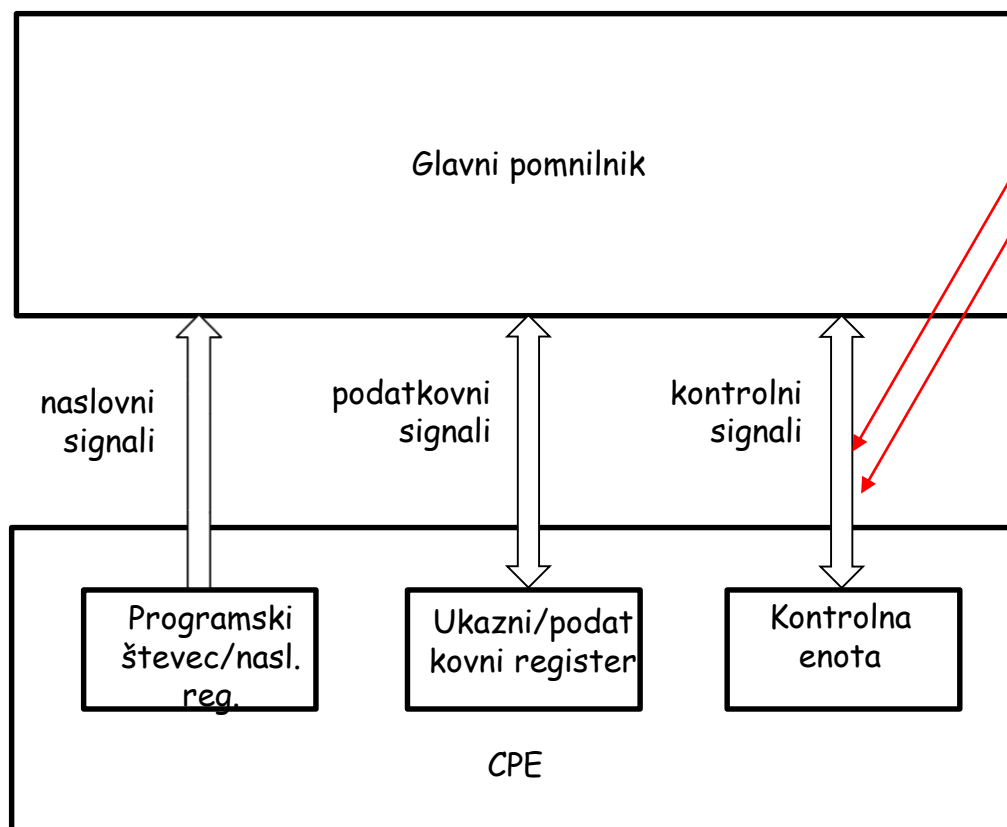
- CPE dostopa do pomnilniške besede tako, da v pomnilnik pošlje naslov te besede in signal za smer prenosa.
- Smer prenosa - vrsta dostopa
 - $CPE \leftarrow$ gl. pomnilnik - branje (bralni dostop)
 - $CPE \rightarrow$ gl. pomnilnik - pisanje (pisalni dostop)



Povezava CPE <-> glavni pomnilnik?

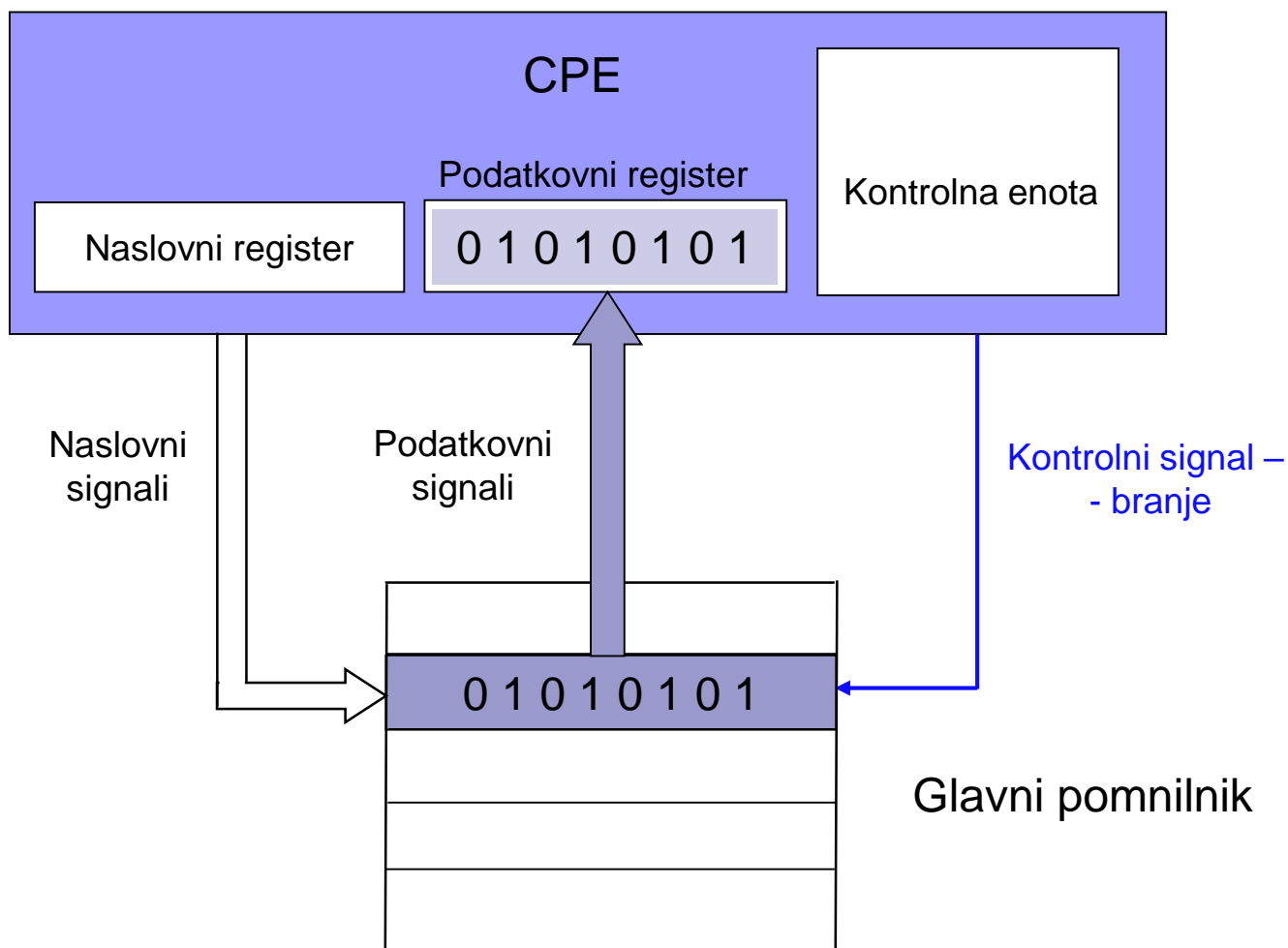
Vodilo = skupina povezav
(naslovno, podatkovno,
kontrolno, ...)

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



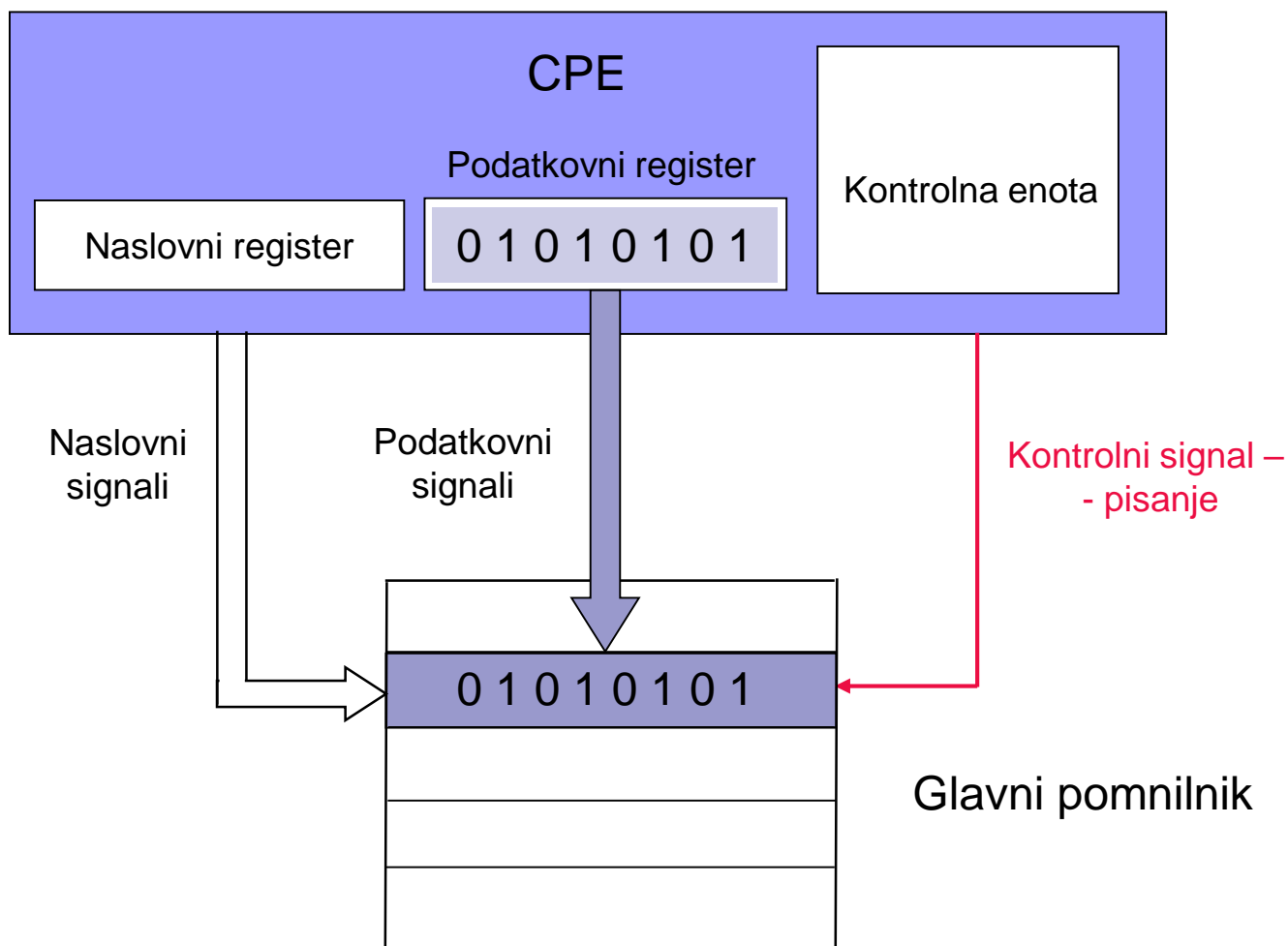


Povezava med CPE in glavnim pomnilnikom – bralni dostop



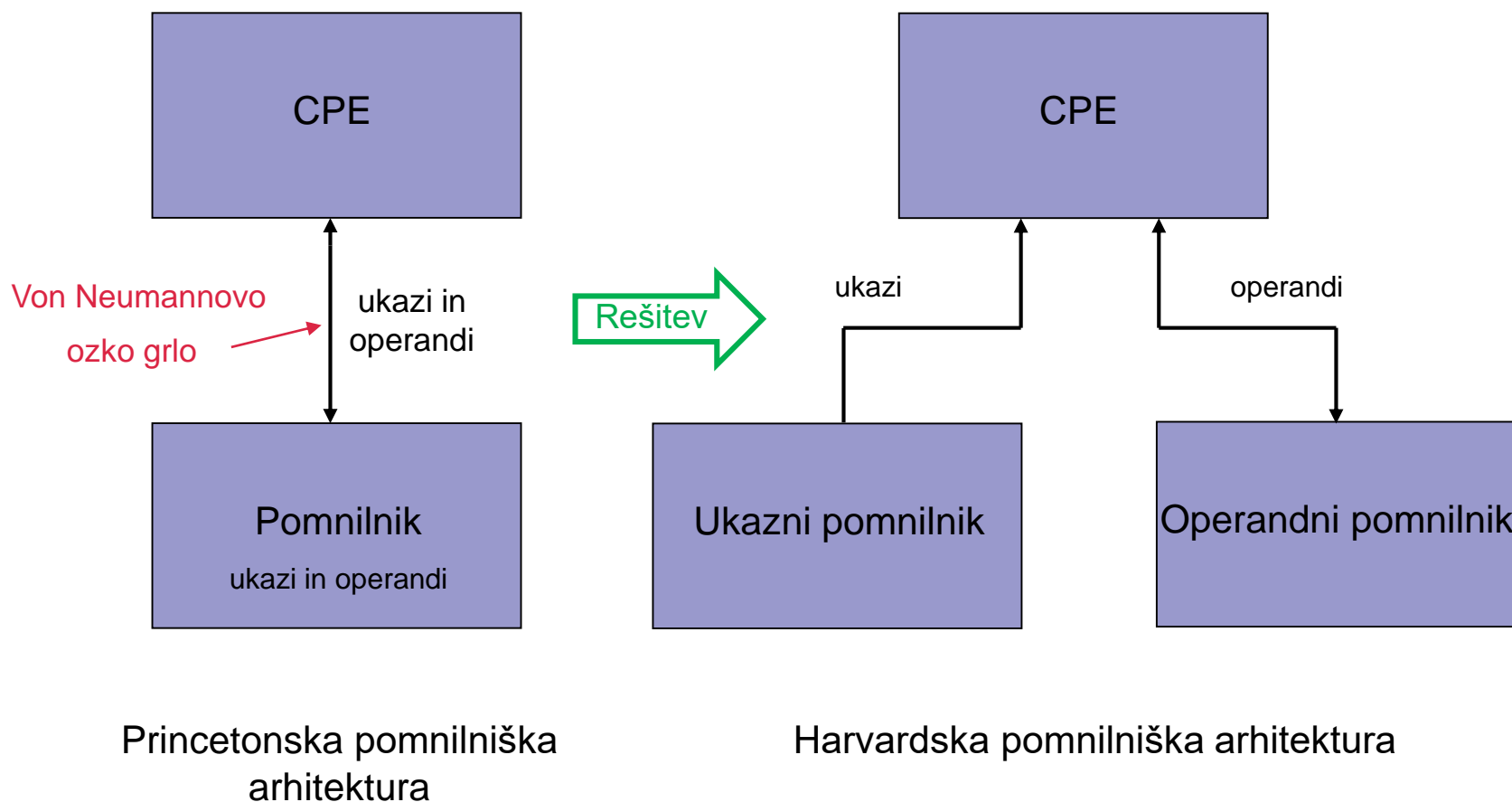


Povezava med CPE in glavnim pomnilnikom – pisalni dostop



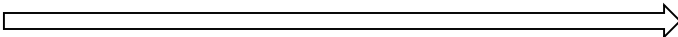


Razširitev von Neumannovega ozkega grla





Povzetek lastnosti glavnega pomnilnika v von Neumannovem računalniku

- Pomnilnik je **enodimenzionalen** in organiziran **v besede**. Vsaka beseda ima svoj, **enoličen naslov**.
- Ni razlike med **ukazi** in **operandi**.
- **Pomen** ni sestavni del operandov.
- **Veliko več bralnih kot pisalnih dostopov**,
 - Razmerje: okrog 80% branj (B), 20% pisanj (P)
 - **Zakaj?** 

Primer programa

PROGRAM	
2B	LDR R1,STEV1
2B	LDR R2,STEV2
1B	ADD R3,R2,R1
1B1P	STR R3,REZ



Kombinacija 8 bitov v pomnilniku,
npr. 1000 1011, lahko predstavlja:

- število brez predznaka: 139 (desetiško)
ali
- število s predznakom: - 11 (desetiško)
ali
- znak v razširjeni ASCII abecedi: <
ali
- strojni ukaz: ADDA (op.koda strojnega ukaza procesorja 68HC11)
ali
- pomnilniški naslov: 139 (desetiško)
ali
- kombinacijo bitov
ali
- točka slike, vzorec zvoka, . . .



Osnove delovanja računalnikov - vsebina:

- ☐ 3.1 Von Neumannov računalniški model
- ☐ 3.2 Flynnova klasifikacija
- ☐ 3.3 Glavni pomnilnik v von Neumannovem računalniku
- ☐ 3.4 Amdahlov zakon
- ☐ 3.5 Jeziki, nivoji in navidezni računalniki
- ☐ 3.6 Primer izvedbe programa v računalniku



3.4 Amdahlov zakon (1967)

- G. M. Amdahl je eden od arhitektov slavne serije računalnikov IBM 370.
- Če v računalniku za faktor N (N -krat) pohitrimo delovanje pri vseh operacijah, razen pri f -temu delu od vseh operacij, potem je povečanje hitrosti celotnega računalnika $S(N)$ enako:

$$S(N) = \frac{1}{f + \frac{1-f}{N}} = \frac{N}{1 + (N-1) * f}$$

f je delež operacij, ki niso pohitrene!

$S(N)$ = povečanje hitrosti celotnega sistema

N = faktor povečanja hitrosti $(1 - f)$ -tega dela

f = delež operacij, ki niso pohitrene

$1 - f$ = delež operacij, ki so N -krat pohitrene



Primer 1:

prej	$f = 0.4$	$1-f = 0.6$
potem	$f = 0.4$	pohitritev 8x ($0.6/8=0.075$)

- Izvajanje programov na nekem računalniku bi želeli pohitriti tako, da enojedrni procesor zamenjamo z osemjedrnim (8 paralelno delujočih CPE).
- Kolikokrat hitreje se bodo izvajali programi, če se lahko paralelno izvaja samo 60 % programov?



Amdahlov zakon

prej	$f = 0.4$	$1-f = 0.6$
potem	$f = 0.4$	pohitritev 8x ($0.6/8=0.075$)

- $N = 8$ (del programov se lahko izvaja 8-krat hitreje)
- $1 - f = 0,6$ delež programov, ki so 8-krat pohitreni;
- $f = 0,4$ delež programov, ki niso pohitreni (40% programov se ne more izvajati paralelno)
- $S(N)$ pohitritev celote (vseh programov)

$$S(N) = \frac{8}{1 + (8 - 1) * 0,4} = \frac{8}{1 + 2,8} = 2,1$$

- Hitrost izvajanja vseh programov se poveča za faktor 2,1 (2,1 - krat).
- Če so se programi pred zamenjavo izvajali npr. 100 sekund, se bodo potem izvajali 47,6 sekunde ($100 / 2,1 = 47,6$).



prej	<table border="1"><tr><td>$f = 0.1$</td><td>$1-f = 0.9$</td></tr></table>	$f = 0.1$	$1-f = 0.9$
$f = 0.1$	$1-f = 0.9$		
potem	<table border="1"><tr><td>$f = 0.1$</td><td>pohitritev 2x (polovični čas)</td></tr></table>	$f = 0.1$	pohitritev 2x (polovični čas)
$f = 0.1$	pohitritev 2x (polovični čas)		

Primer 2:

- Izvajanje programa na nekem računalniku bi želeli pohitriti tako, da izvajanje 90% ukazov v programu dvakrat pohitrimo.
- Kolikokrat hitreje se bo izvajal program na takem računalniku ?

$$S(N) = ?$$



prej

$f = 0.1$	$1-f = 0.9$
-----------	-------------

Primer 2:

potem

$f = 0.1$	pohitritev 2x (polovični čas)
-----------	-------------------------------

- Izvajanje programa na nekem računalniku bi želeli pohitriti tako, da izvajanje 90% ukazov dvakrat pohitrimo.
- Kolikokrat hitreje se bo izvajal program na takem računalniku ?

$$S(N) = \frac{1}{0.1 + \frac{0.9}{2}} = \frac{1}{0.1 + 0.45} = \frac{1}{0.55} = 1.818181$$

- Hitrost izvajanja programa se poveča za faktor 1.82 .



Osnove delovanja računalnikov - vsebina:

- ☐ 3.1 Von Neumannov računalniški model
- ☐ 3.2 Flynnova klasifikacija
- ☐ 3.3 Glavni pomnilnik v von Neumannovem računalniku
- ☐ 3.4 Amdahlov zakon
- ☐ 3.5 Jeziki, nivoji in navidezni računalniki
- ☐ 3.6 Primer izvedbe programa v računalniku



3.5 Jeziki, nivoji in navidezni računalniki

- Za veliko večino uporabnikov so podrobnosti o zgradbi in delovanju računalnikov nepomembne.
- Računalnik in njegove lastnosti vidijo predvsem skozi lastnosti programskega jezika, ki ga uporabljajo.
- Neki programski jezik se lahko realizira na zelo različnih računalnikih, to pa pomeni, da so različni računalniki za uporabnika, ki uporablja ta programski jezik, videti bolj ali manj enaki.

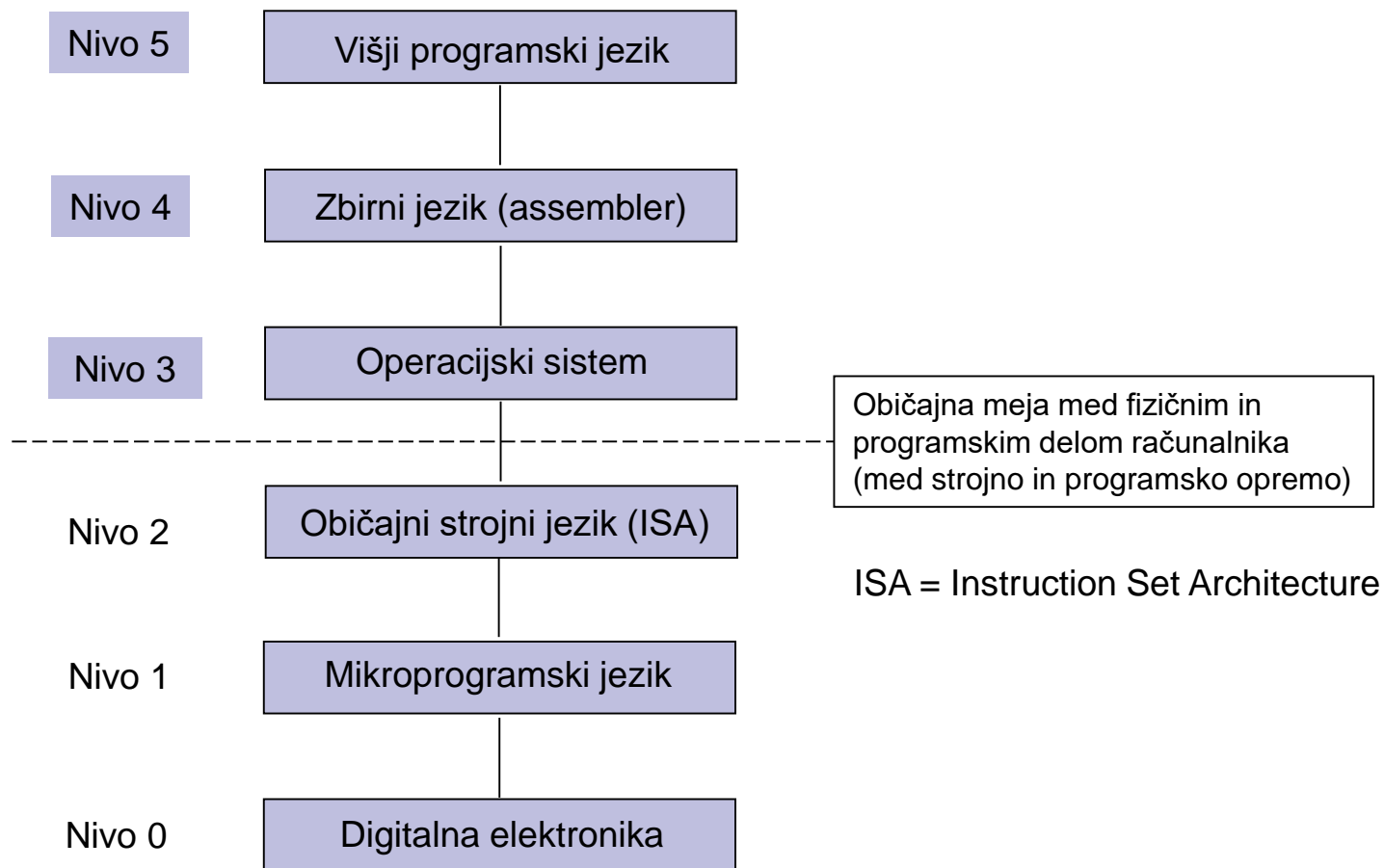


Računalnik kot zaporedje navideznih računalnikov

- Pri veliki večini današnjih računalnikov imamo 6 nivojev.
- Na vsakem nivoju vidimo računalnik skozi drugačen programski jezik.
- Ta programski jezik si lahko uporabnik predstavlja kot strojni jezik nekega navideznega računalnika.
- Na najnižjem nivoju (nivo 0) elektronika (logična vrata in flip-flopi) neposredno izvaja najenostavnejše ukaze.



Računalnik s šestimi nivoji





- Nivo 1 lahko zasledimo pri mnogih današnjih računalnikih. RISC računalniki nimajo nivoja 1.
 - Vsak ukaz običajnega strojnega jezika se izvrši kot zaporedje mikroukazov – računalnikom, ki tako delujejo (imajo nivo 1), rečemo, da so mikroprogramirani.
 - Pri teh računalnikih je mikroprogramski jezik dejansko pravi strojni jezik.
 - Ker v začetku računalniki tega nivoja niso imeli in je za uporabnika neviden, se pojem strojni jezik uporablja na nivoju 2.
 - Mikroprogram na nivoju 1 je napisan pri proizvajalcu in pravzaprav definira običajni strojni jezik. Uporabnik ga običajno ne more spreminjati.



- Uporabnik vidi računalnik na nivoju 2 skozi uporabo običajnih strojnih ukazov, ki tvorijo običajni strojni jezik.
 - Računalniška arhitektura je določena z zgradbo in lastnostmi računalnika, kot jih vidi programer na tem nivoju.
 - Zato tudi ime ISA – Instruction Set Architecture.
 - Z običajnim strojnim jezikom ima programer popoln nadzor nad vsemi deli računalnika.
 - Pri prvih računalnikih višjih nivojev sploh ni bilo in je programiranje potekalo samo v običajnem strojnem jeziku.



- Nivo 3 je nivo operacijskega sistema.
 - Jezik na tem nivoju vsebuje vse ukaze nivoja 2, ki so jim dodani novi ukazi za lažje delo z računalnikom (npr. delo z V/I napravami, paralelno izvajanje programov, diagnostični ukazi).
 - Operacijski sistem je program, ki olajša delo z računalnikom in služi kot vmesnik med uporabnikom in strojno opremo računalnika.
 - Z operacijskim sistemom želimo doseči:
 - lažje delo,
 - boljši izkoristek strojnih zmogljivosti računalnika (v določenem času opraviti kar največ dela).



- Funkcije operacijskega sistema bi bilo mogoče realizirati tudi strojno na nivoju 2, vendar je programska izvedba bolj ekonomična (več operacijskih sistemov, nadgradnja . . .).
- Na tem nivoju je običajna tudi delitev uporabnikov z različno pravico uporabe ukazov.
- Nekateri ukazi nivoja 2 so običajnim uporabnikom na nivoju 3 nedostopni (dostopni samo sistemskim programerjem).
- Za večino današnjih programerjev je nivo 3 najnižji nivo, na katerem lahko delajo.



- Na nivoju 4 uporabnik vidi računalnik skozi zbirni jezik.
 - Zbirni jezik je samo simbolična, človeku bližja oblika jezika nivoja 3 (in s tem tudi nivoja 2).
 - Programe v zbirnem jeziku je treba pred izvajanjem prevesti na jezik nivoja 3 (oziroma 2).

- Nivo 5 oblikujejo višji programski jeziki, ki so namenjeni večini programerjev.
 - To so npr. C, C#, C++, Java, Python, BASIC, FORTRAN, COBOL in mnogi drugi.
 - Programe, napisane v teh jezikih, je treba prevesti na jezik nivoja 4 ali nivoja 3.



- V računalnikih lahko ugotovimo tudi višje nivoje, kot npr. program za delo s podatkovnimi bazami, UI,
- Vsak nivo si lahko predstavljamo kot navidezni računalnik, ki ima za strojni jezik kar jezik tega nivoja, tako da običajnemu uporabniku na višjih nivojih ni potrebno poznavanje dejanskega strojnega nivoja.
- Vsekakor pa je potrebno programe, napisane v jeziku kateregakoli višjega nivoja (navideznega računalnika), pretvoriti v zaporedje ukazov strojnega jezika.
- Uporabniki se tega pretvarjanja pogosto ne zavedajo, proizvajalci računalnikov in programske opreme pa morajo poskrbeti za prehajanje iz enega jezika v drugega.



- Mehanizem prehajanja iz enega jezika v drugega je lahko realiziran na dva načina:
 - ☐ s prevajanjem,
 - ☐ z interpretiranjem.

- Po letu 1990 pa se je razširila še vmesna rešitev:
 - ☐ delno prevajanje.

- Glavna razlika med prevajanjem in interpretiranjem je, da pri interpretiranju ne obstaja prevedeni (ciljni) program.

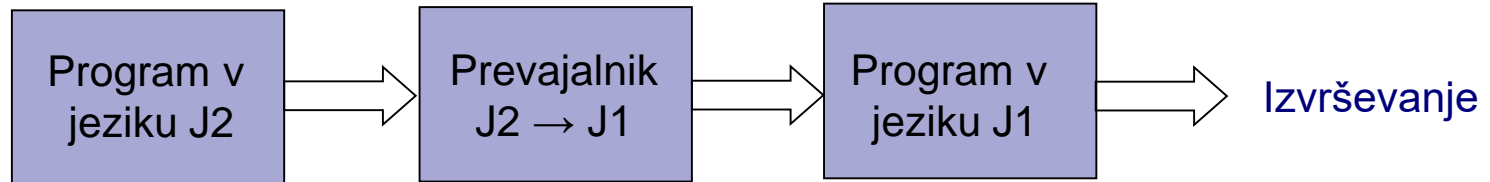


Prehajanje iz jezika J2 v jezik J1

Prevajanje

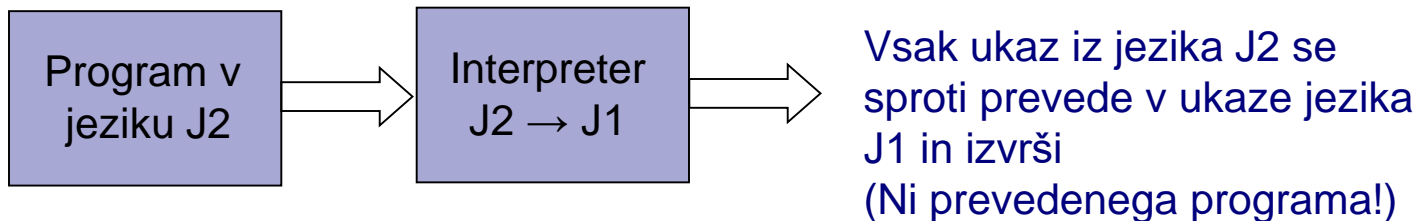
Izvorni program

Prevedeni program



Interpretiranje

Izvorni program





- ☐ Prevedeni programi delujejo samo na računalniku s strojnim jezikom, v katerega so bili prevedeni.
- ☐ Pred prenosom na drugačen računalnik (z drugačnim strojnim jezikom J1) je treba izvorni program znova prevesti.
- Z vključevanjem velikega števila različnih računalnikov v omrežja je postala prenosljivost programov, ki jo omogoča interpretiranje, zelo pomembna.
- Delno prevajanje je neka vmesna rešitev med interpretiranjem in prevajanjem, ki omogoča hitrejša interpretiranje.



- Delno prevajanje: Izvorni program v jeziku J2 se prevede v program v vmesnem jeziku J1, program v J1 pa se interpretira.
- Delno prevajanje v vmesni jezik J1 omogoča hitrejšo interpretiranje, ki pa je vseeno tipično 10-krat počasnejše kot izvajanje v celoti prevedenega programa.
- Tako je omogočena prenosljivost programov pri bistveno manjši izgubi hitrosti, kot če bi uporabili samo interpretiranje.

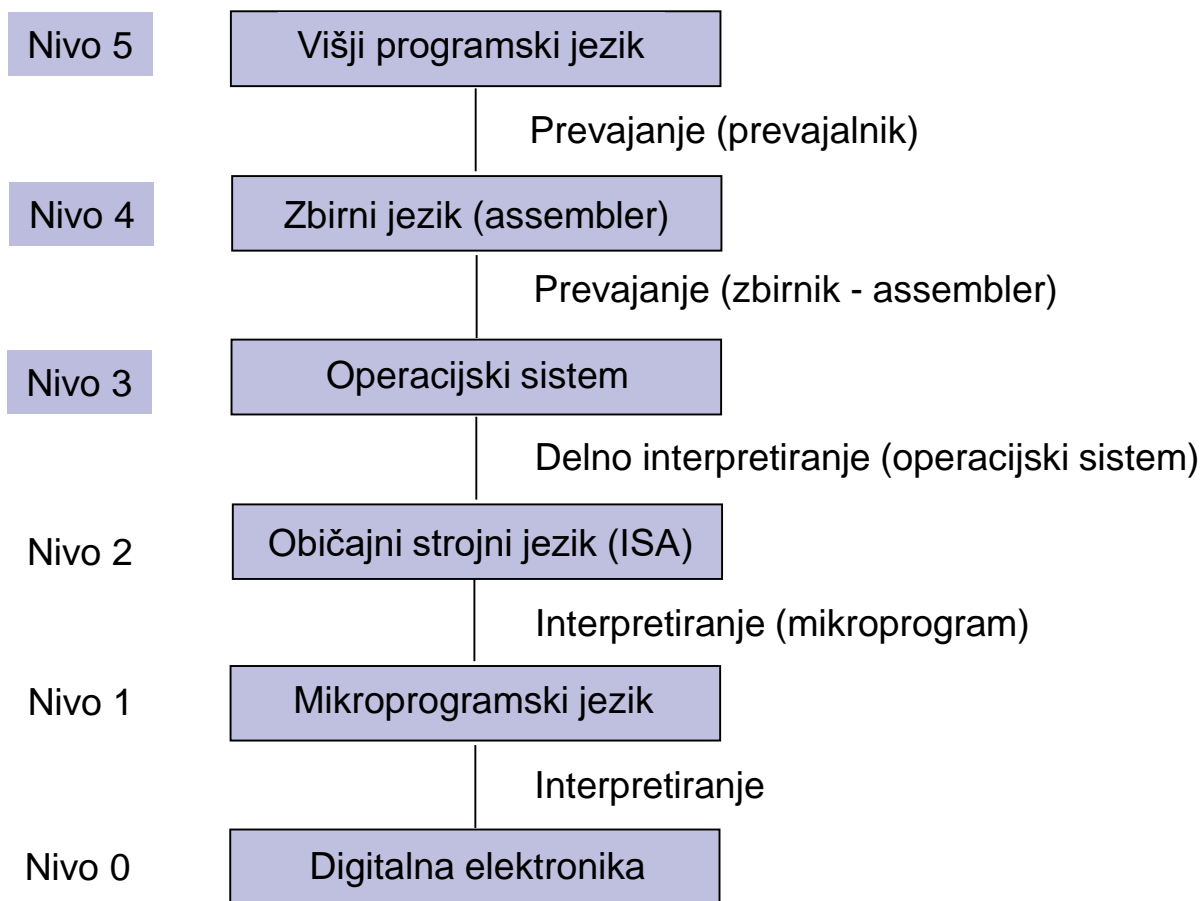


■ Primer navideznega računalnika: JVM (Java Virtual Machine)

- Virtual Machine – navidezni stroj (navidezni računalnik) je programska izvedba stroja (računalnika), ki deluje (izvaja programe) enako kot realen stroj (računalnik).
- Javanski programi se izvajajo tako, da se najprej prevedejo (delno prevajanje) v neki vmesni jezik (Java byte code), ki se interpretira s programom JVM.

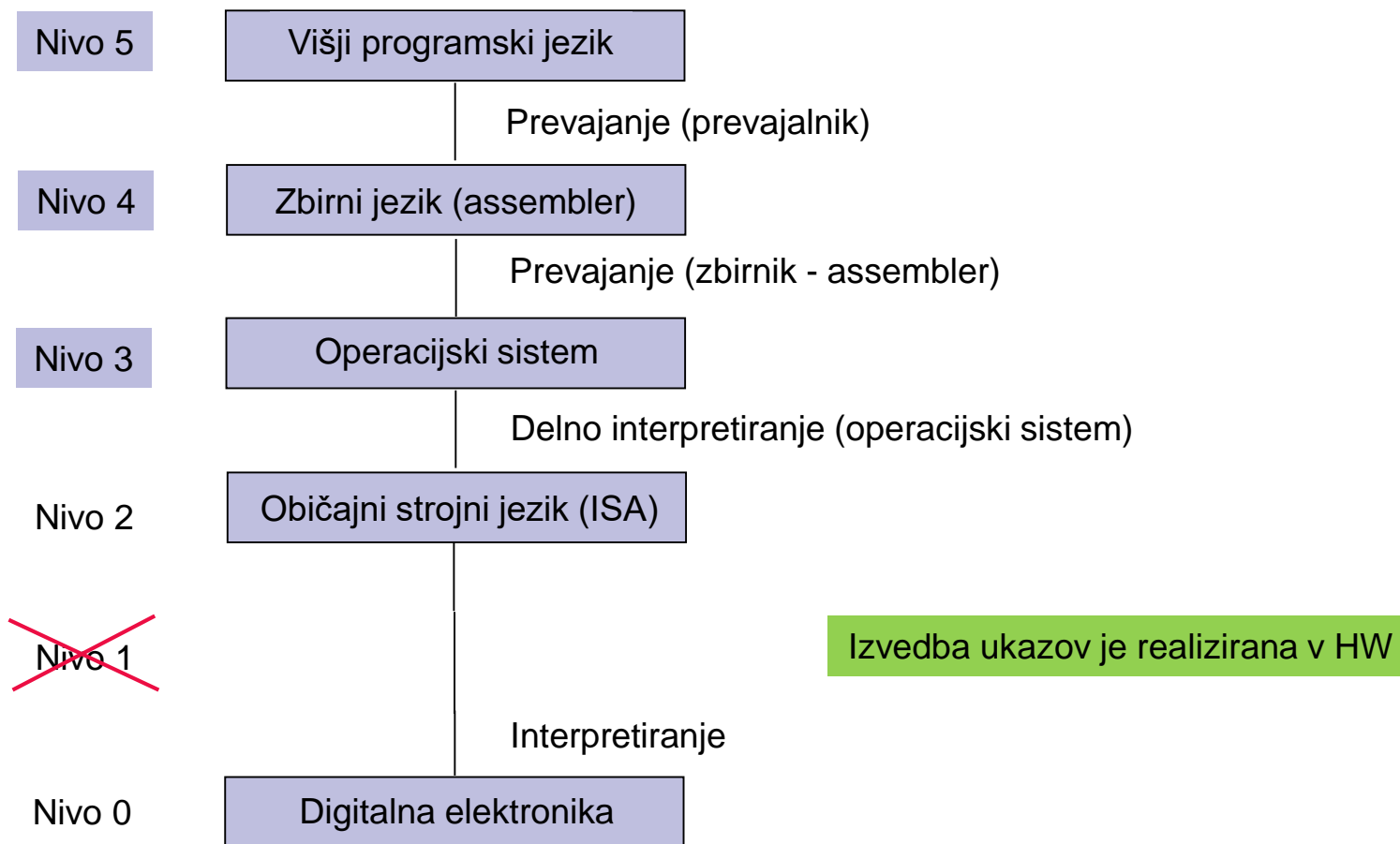


Računalnik s šestimi nivoji (mikroprogramiran)





Računalnik s petimi nivoji (RISC računalnik)





Strojna in programska oprema računalnika

- Meja med strojnim in programskim delom računalnika ni trdna - lahko jo premikamo.
- Vsakega od nivojev lahko realiziramo tako strojno kot tudi programsko.
- Nivo 2 je npr. lahko realiziran s programom, ki teče na drugem računalniku.

Strojna in programska oprema sta logično ekvivalentni.



- Vsaka operacija, ki jo izvede programska oprema, se lahko realizira tudi direktno strojno (hardversko).
- Prav tako pa vsak strojni ukaz, ki ga izvaja hardver, lahko simuliramo s programom.
- Razvoj večnivojskih strojev
 - Iznajdba mikroprogramiranja (1951)
 - Iznajdba operacijskega sistema (okrog 1960)
 - Selitev funkcionalnosti v mikroprogram (okrog 1970)
 - Opuščanje mikroprogramiranja (po 1984)
 - Danes običajno kombinacija:
 - kompleksni ukazi običajnega strojnega nivoja so realizirani mikroprogramsko, enostavnejši ukazi so realizirani strojno.



Osnove delovanja računalnikov - vsebina:

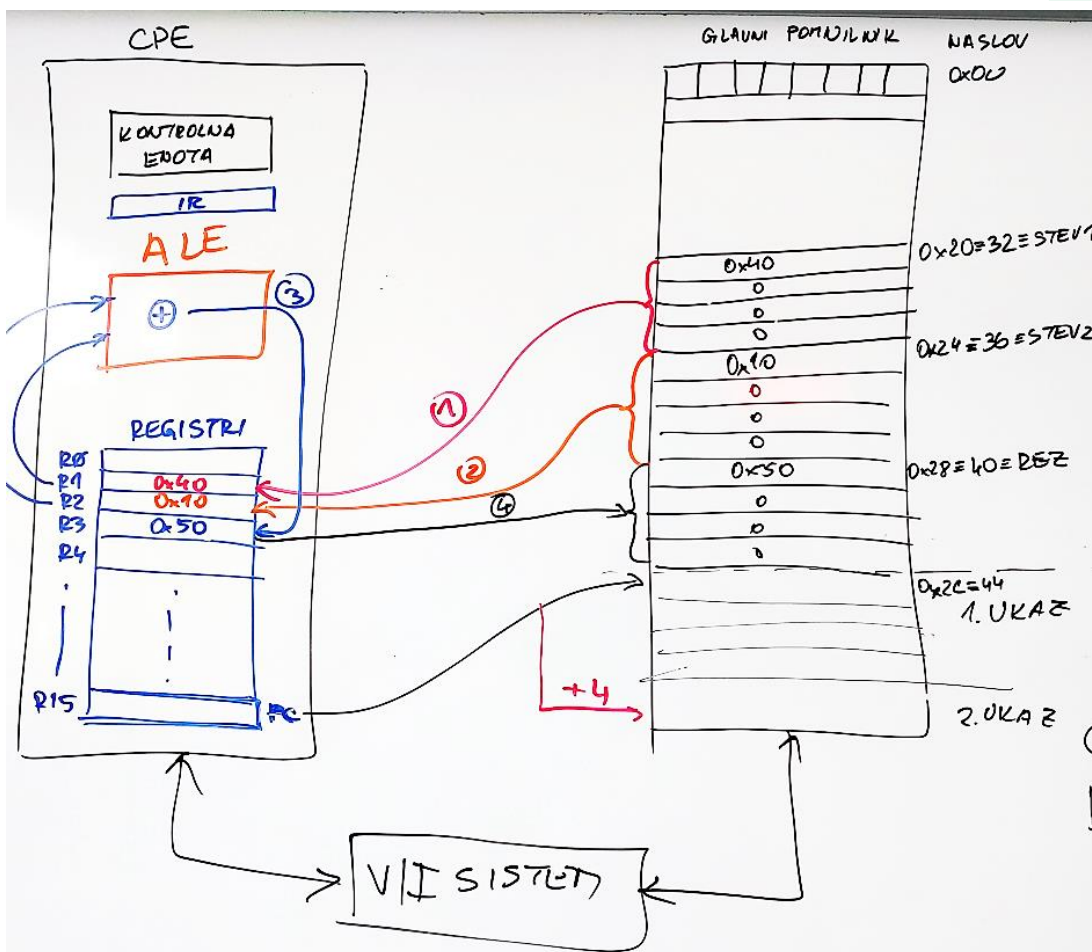
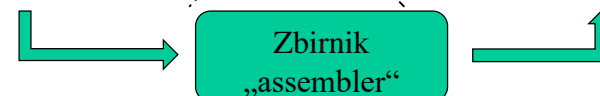
- ☐ 3.1 Von Neumannov računalniški model
- ☐ 3.2 Flynnova klasifikacija
- ☐ 3.3 Glavni pomnilnik v von Neumannovem računalniku
- ☐ 3.4 Amdahlov zakon
- ☐ 3.5 Jeziki, nivoji in navidezni računalniki
- ☐ 3.6 Primer izvedbe programa v računalniku



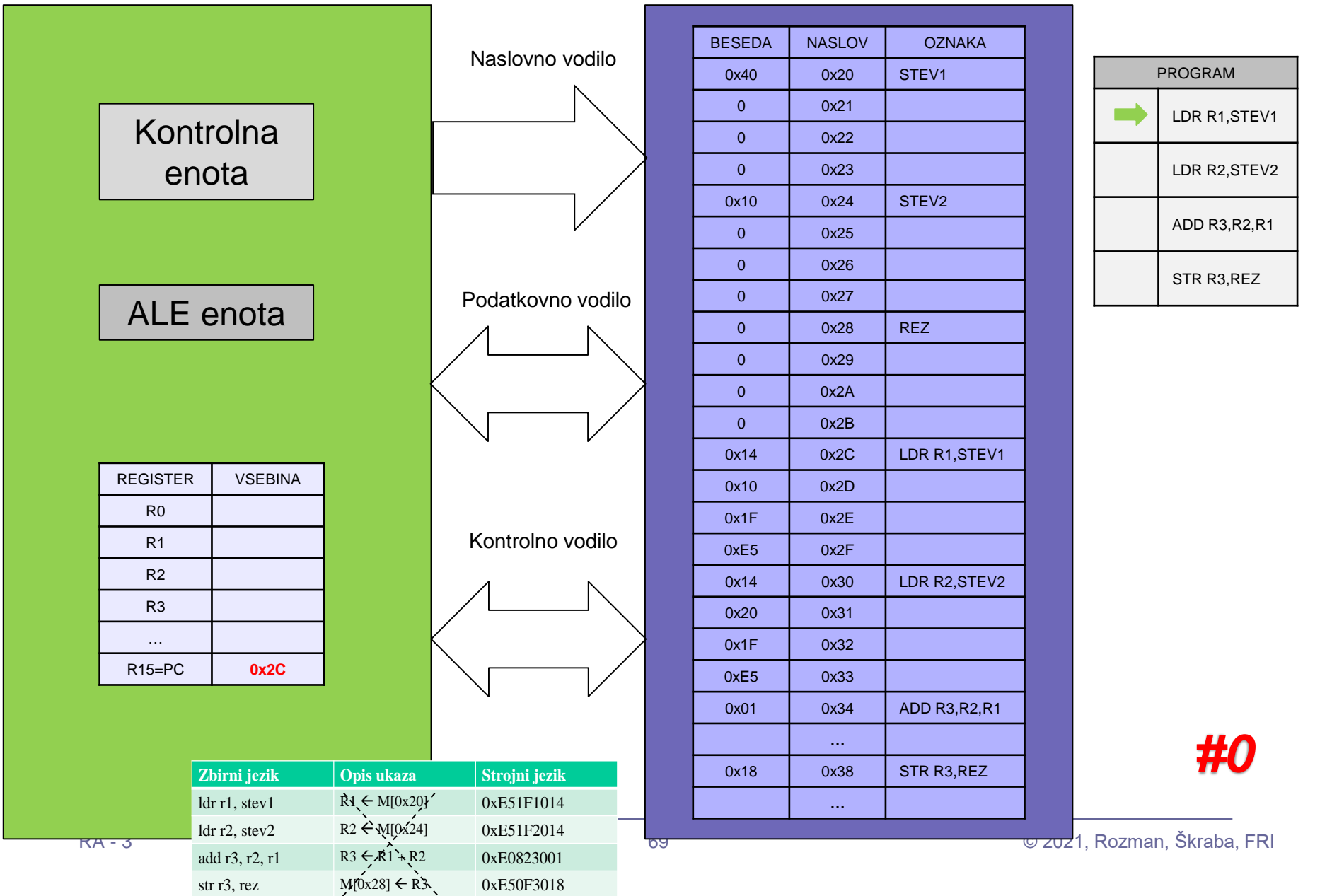
3.6 Primer izvedbe programa v računalniku

Zgled seštevanja dveh števil :
rez := stev1 + stev2

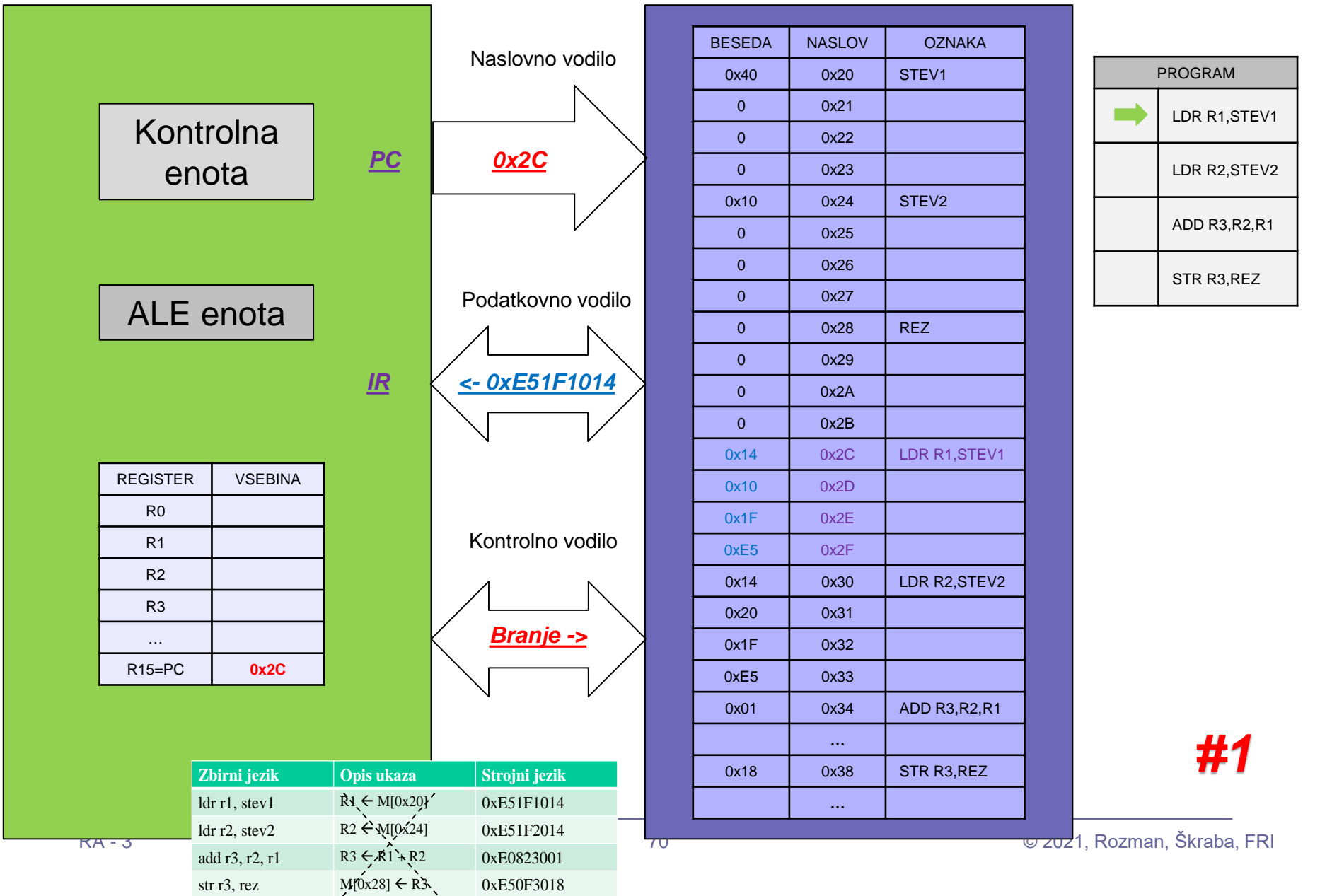
Zbirni jezik	Opis ukaza	Strojni jezik
ldr r1, stev1	$R1 \leftarrow M[0x20]$	0xE51F1014
ldr r2, stev2	$R2 \leftarrow M[0x24]$	0xE51F2014
add r3, r2, r1	$R3 \leftarrow R1 + R2$	0xE0823001
str r3, rez	$M[0x28] \leftarrow R3$	0xE50F3018

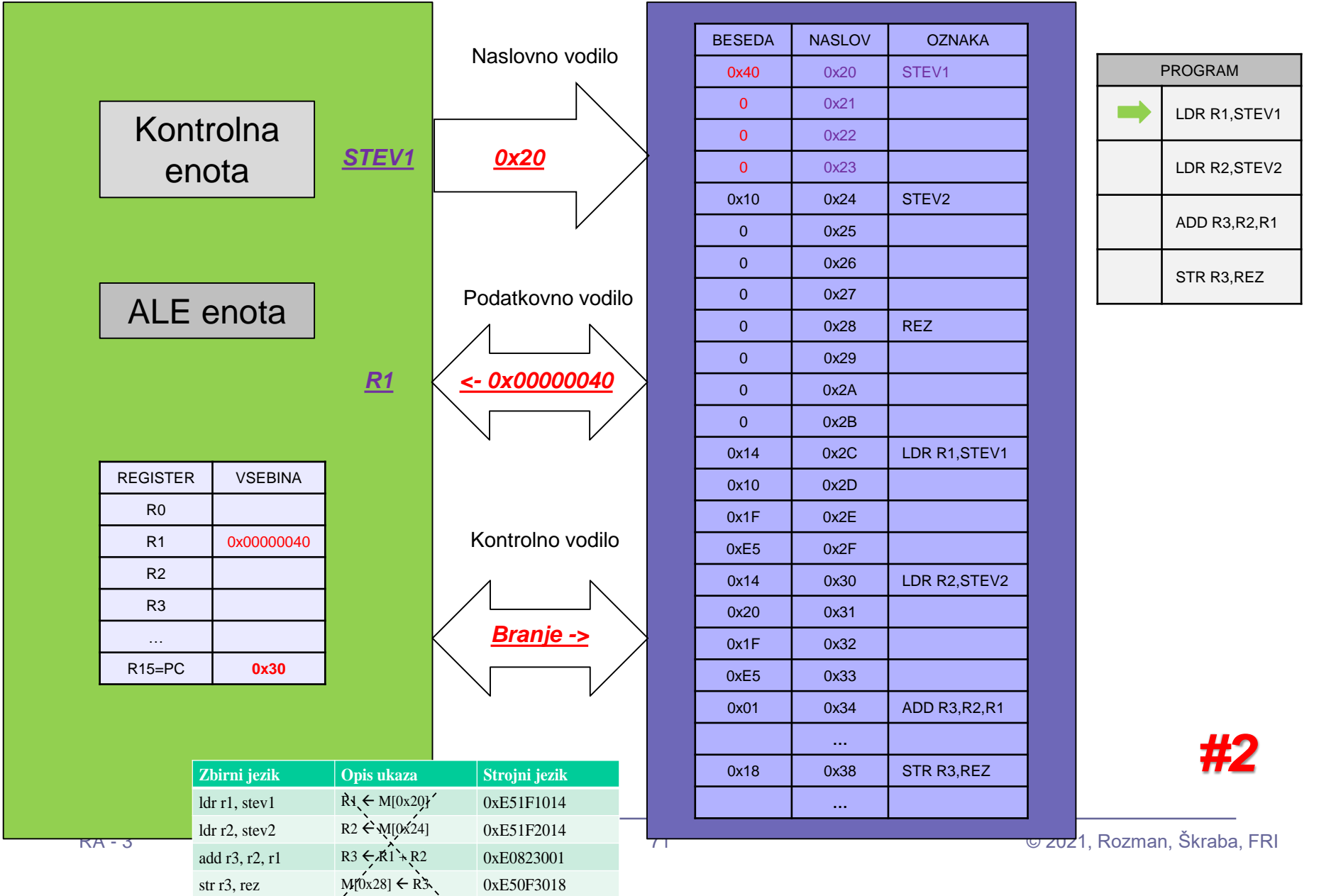


Primer izvedbe programa



Primer izvedbe programa

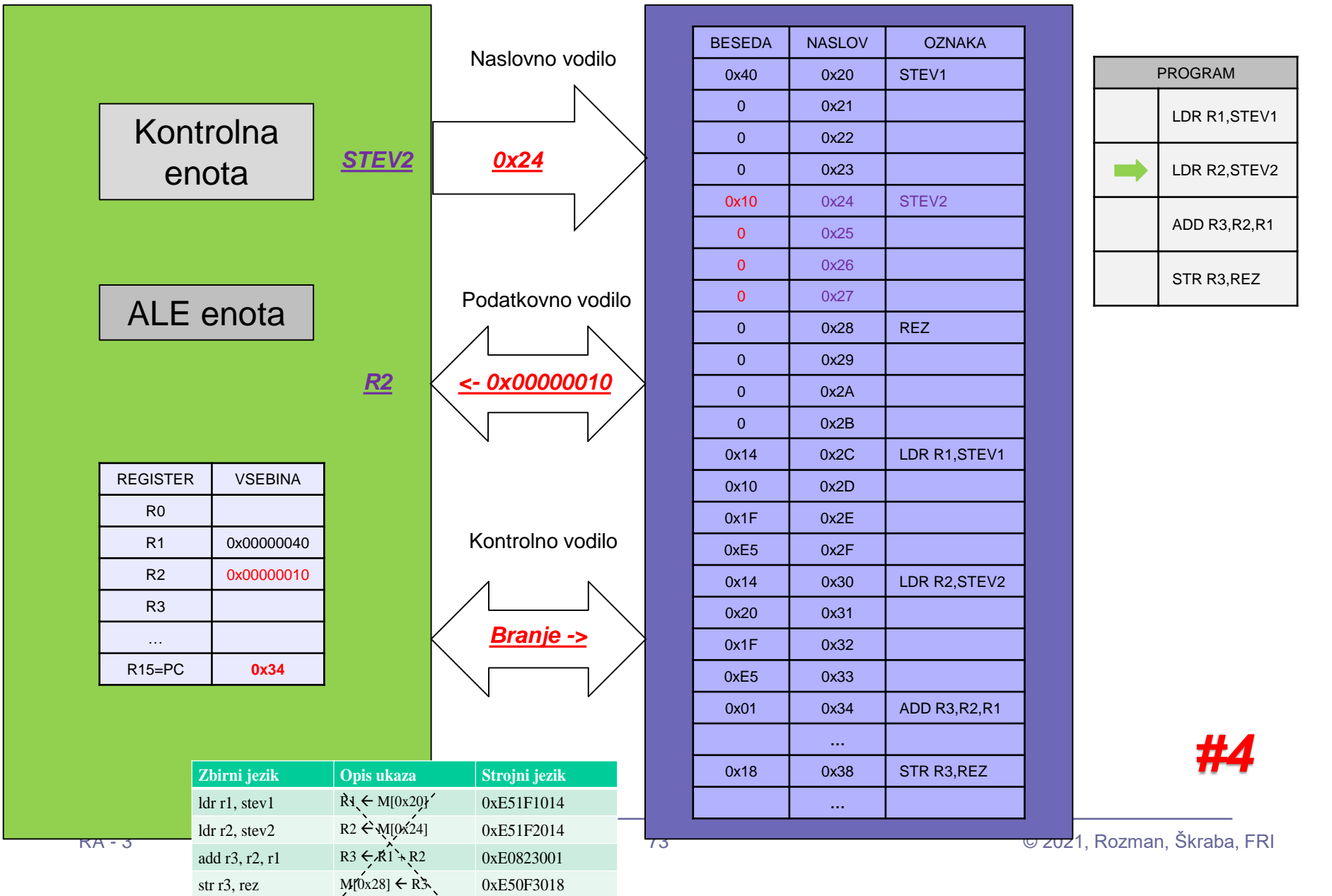




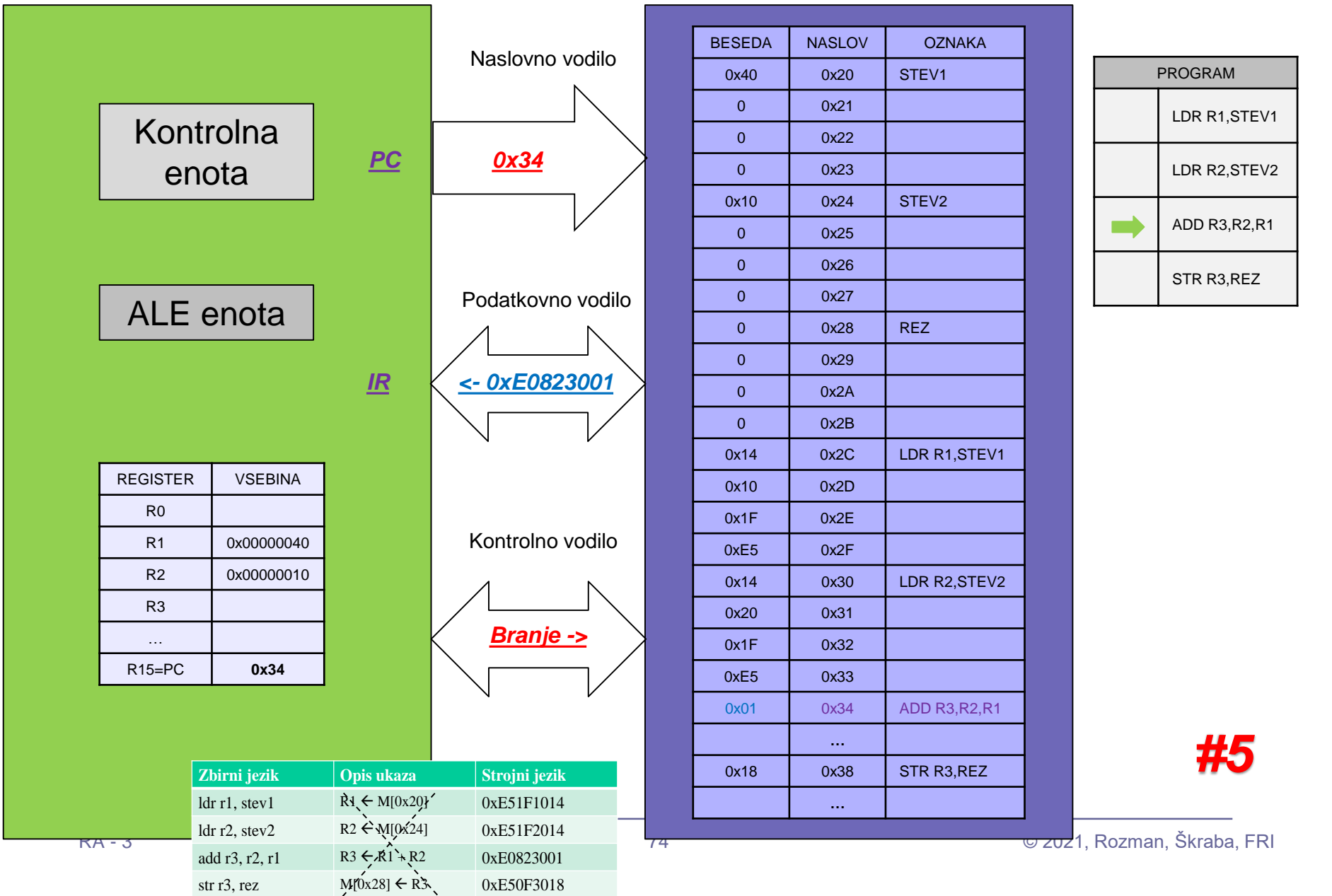
RA - 3



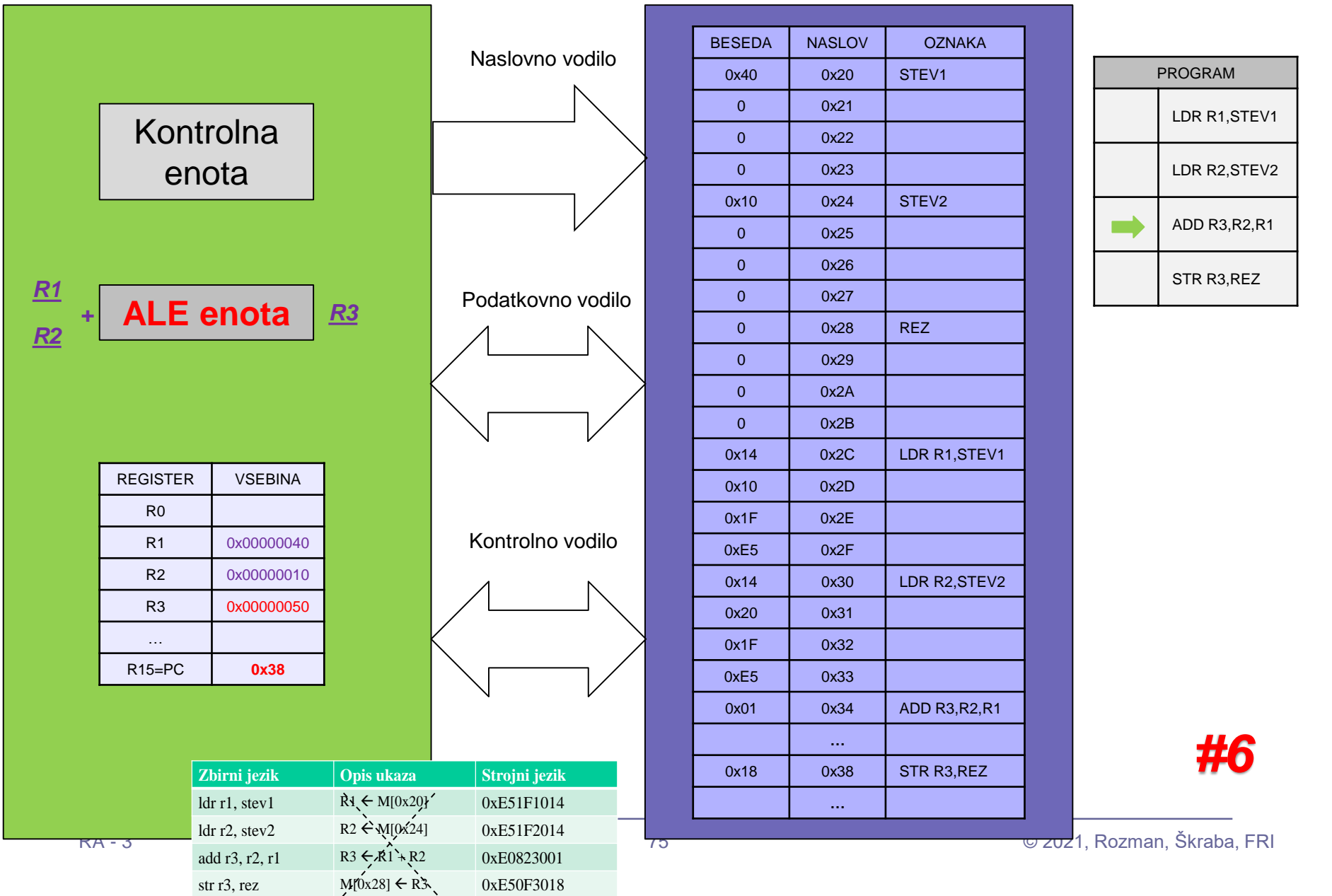
Primer izvedbe programa



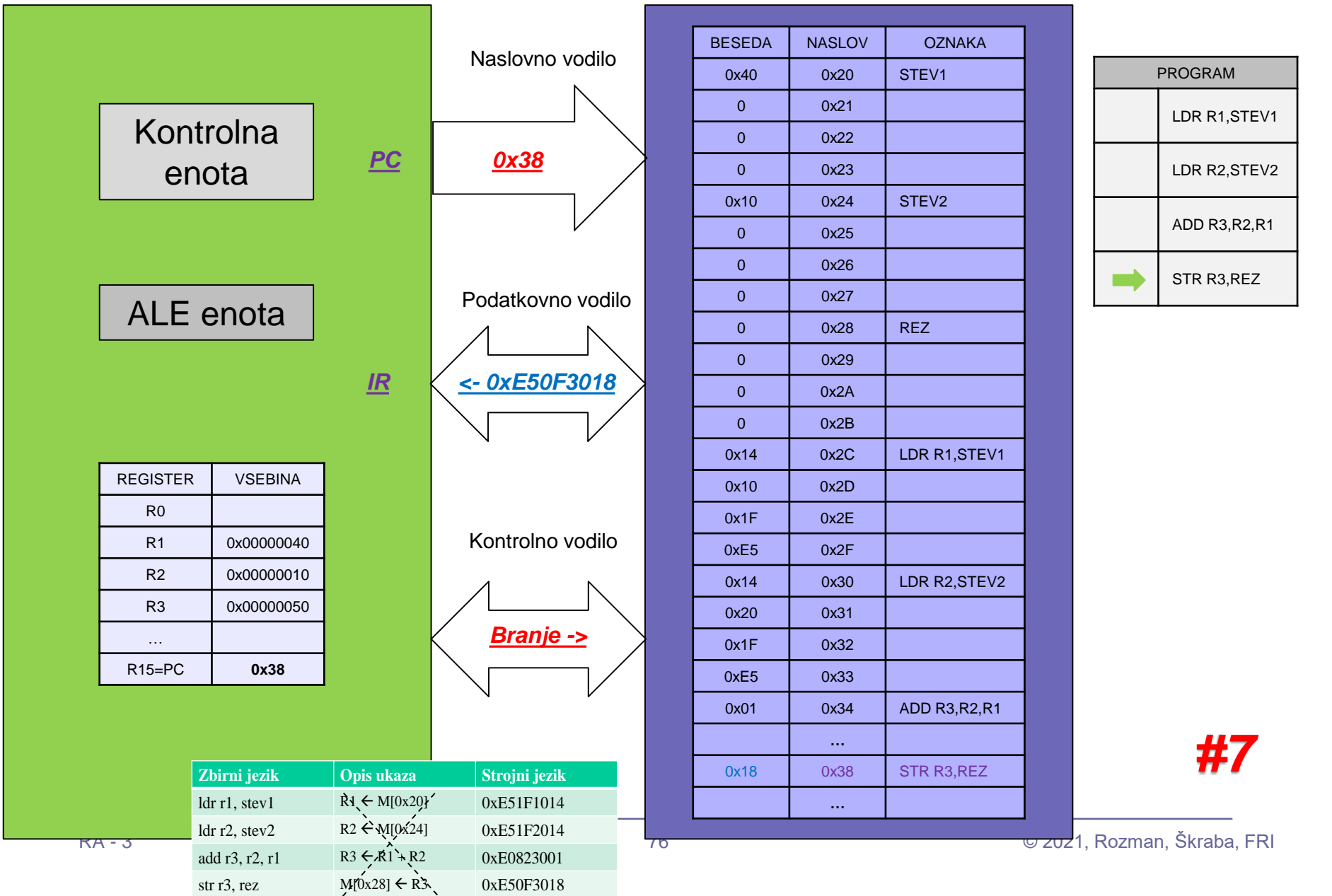
Primer izvedbe programa



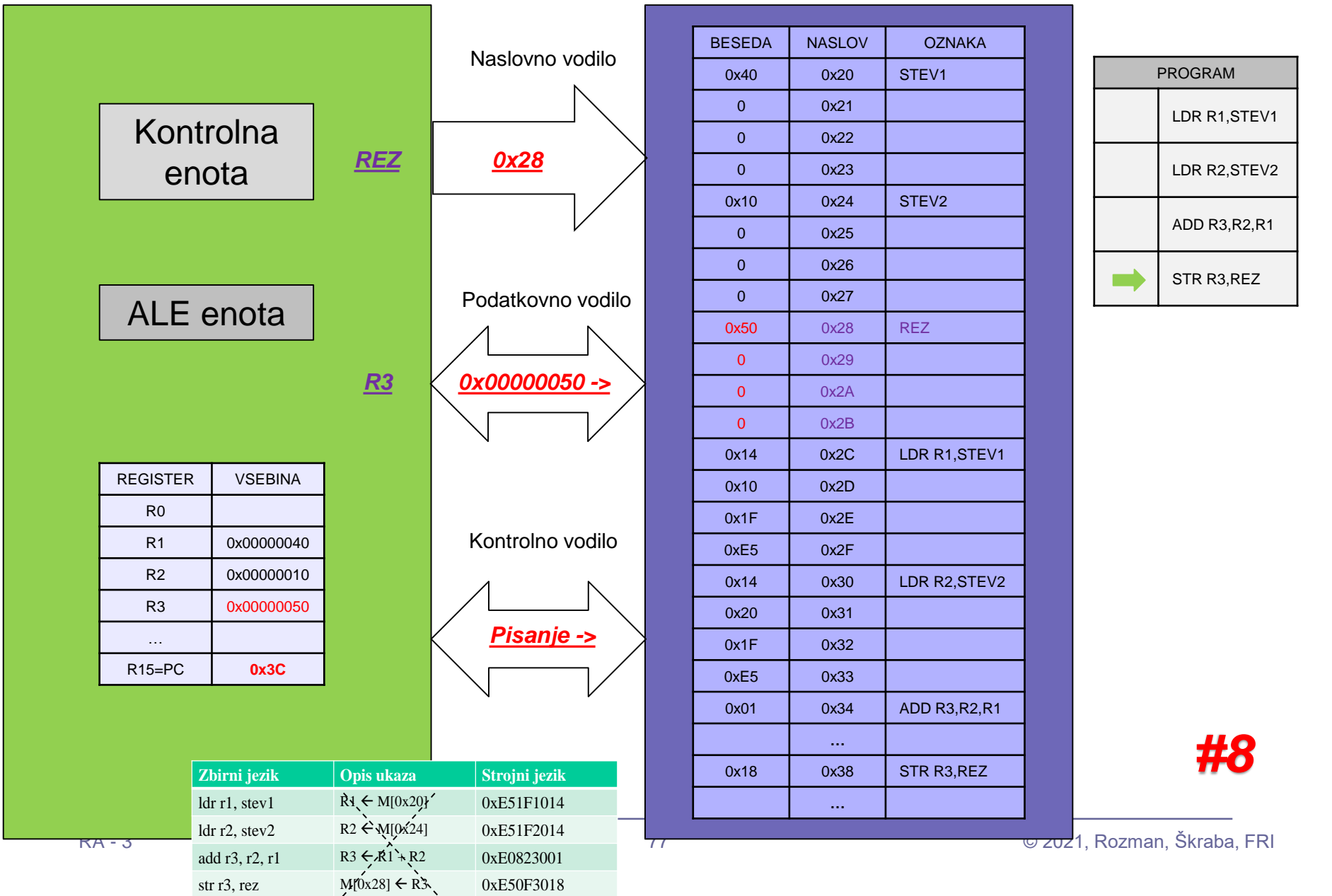
Primer izvedbe programa



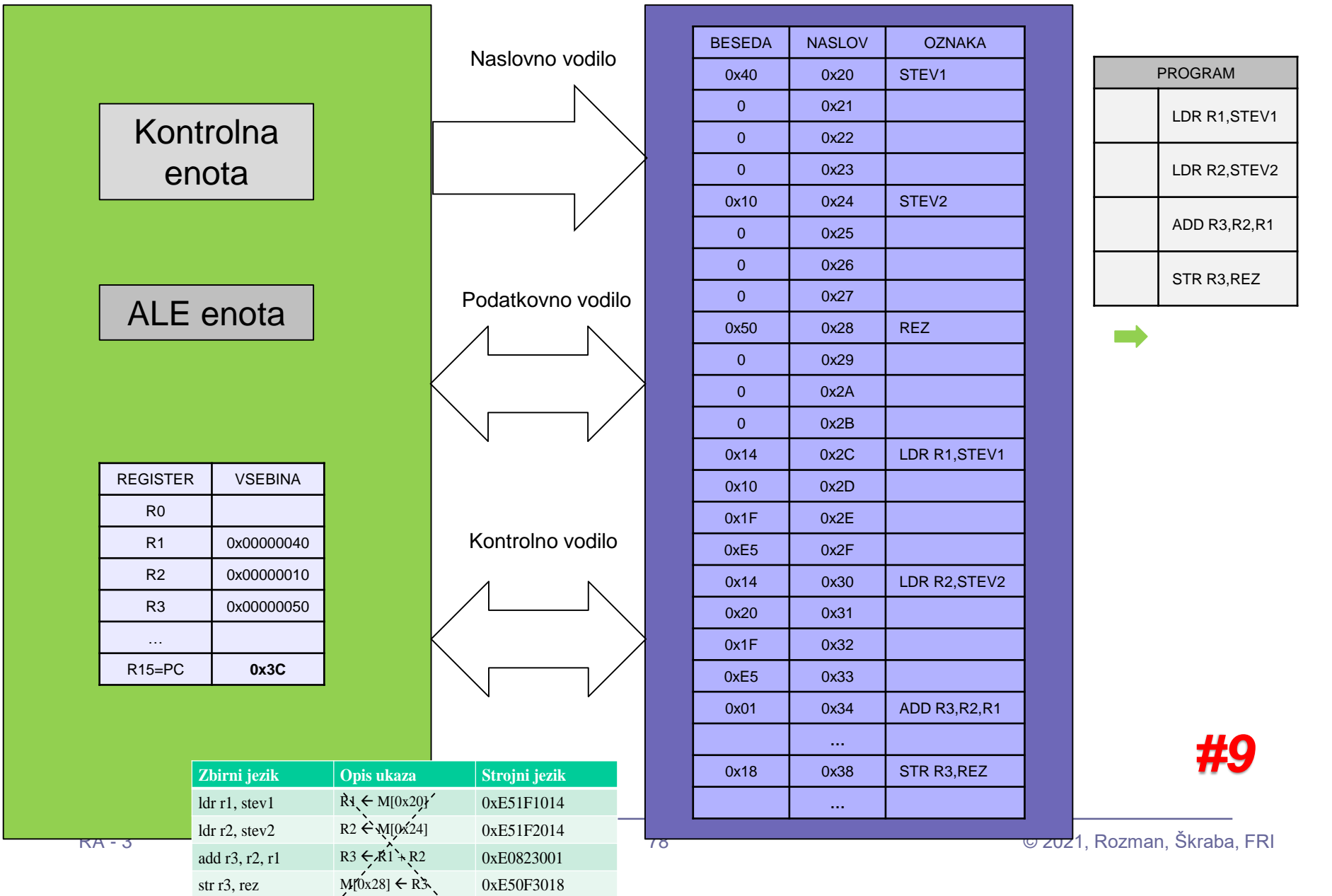
Primer izvedbe programa



Primer izvedbe programa



Primer izvedbe programa



Primer izvedbe programa: Tabela izvedbe (izpolnite za vajo)

[illegible]