RAČUNALNIŠKA ARHITEKTURA

6 Centralna procesna enota - CPE

6 Centralna procesna enota - cilji:

- Osnovno razumevanje :
  - zgradbe (osnovna elektronska vezja) in delovanja CPE
  - sinhronizacije delovanja vezij z urinim signalom
  - mikroprogramska (SW) ali trdoožičena (HW) izvedba CPE
- Razumevanje paralelizacije:
  - vzroki obstoja
  - izvedba paralelizacije na nivoju ukazov
    - cevovod
- Splošno razumevanje izvedbe ukazov v CPE

6 Centralna procesna enota - vsebina:

6.1 Osnove zgradbe in delovanja CPE

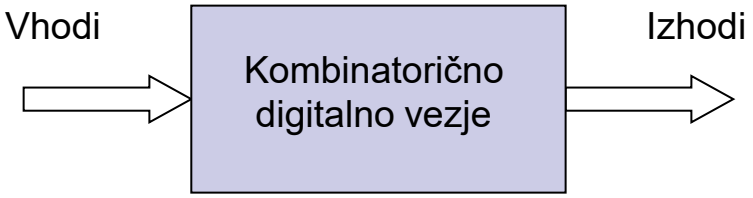
- CPE (Centralna procesna enota ali tudi procesor) je enota, ki izvršuje ukaze, zato njena zmogljivost v veliki meri določa zmogljivost računalnika.

- Poleg CPE ima večina računalnikov še druge procesorje, večinoma v vhodno/izhodnem delu računalnika.

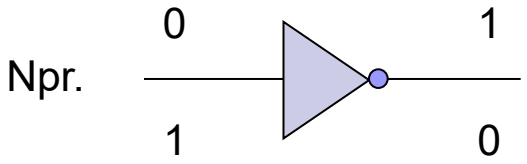
- Osnovni principi delovanja so za vse vrste procesorjev enaki.

Kombinatorična digitalna vezja

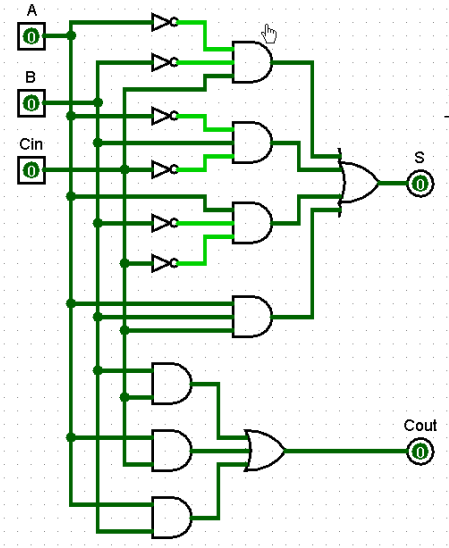
- CPE je digitalni sistem (zgrajena iz digitalnih elektronskih vezij) posebne vrste.
- Dve skupini digitalnih vezij:
  - Kombinatorična digitalna vezja
    - Stanje izhodov je odvisno samo od trenutnega stanja vhodov



Primer: Negator



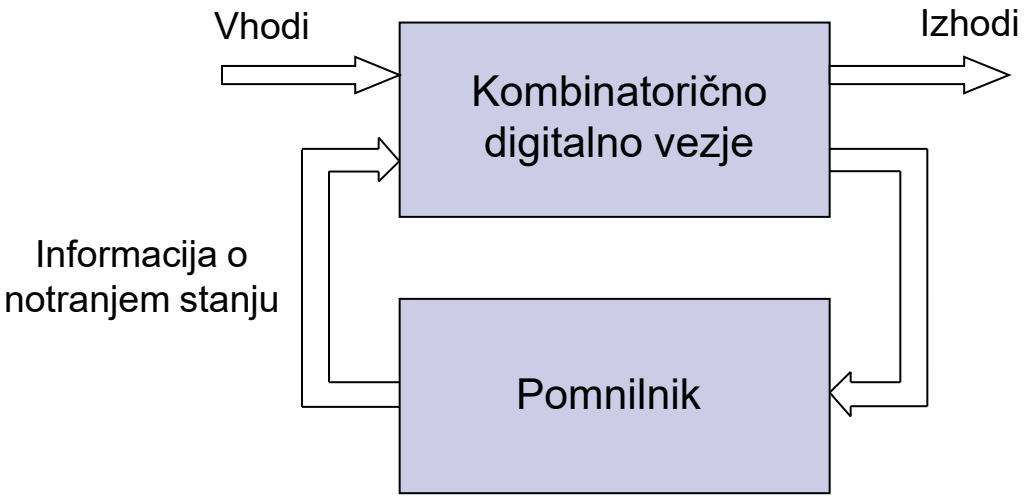
Primer: 1-bitni seštevalnik



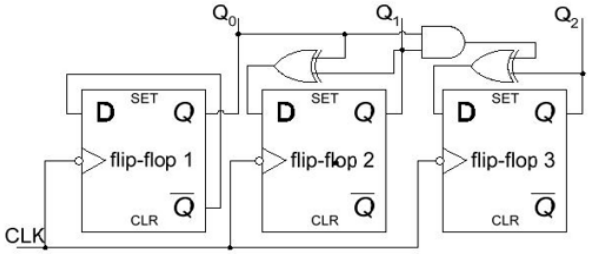
x_i	y_i	c_i	s_i	c_{i+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Pomnilniška (sekvenčna) digitalna vezja

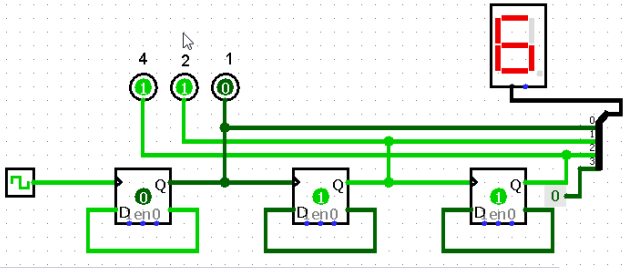
- Stanje izhodov je odvisno od trenutnega stanja vhodov in tudi od prejšnjih stanj vhodov
- Pomnilniška vezja si zapomnijo stanja
- Prejšnja stanja običajno označimo kot **notranja stanja**, ki odražajo prejšnja stanja vhodov



Primer: 3-bitni števec

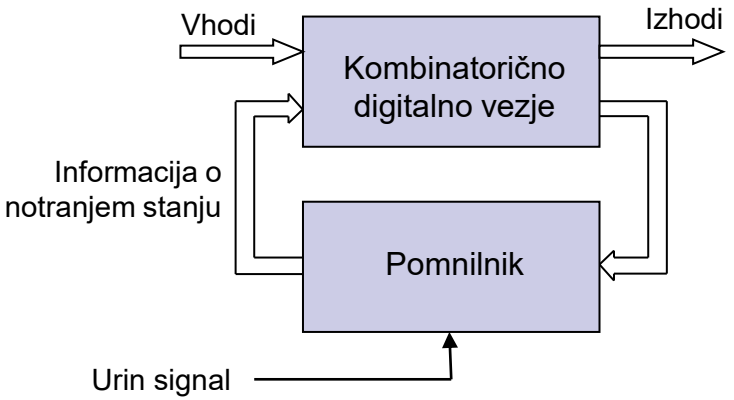


Primer: 3-bitni števec - Logisim



Pomnilniška (sekvenčna) vezja

- Flip-flop – enobitna pomnilniška celica
- Register
- Števec
- Pomnilnik



- Pomnilniška (sekvenčna) digitalna vezja so lahko:
  - Asinhronska - stanje vezja se spremeni „takoj“ ob spremembi vhodnih signalov.
  - Sinhronska - stanje vezja se v odvisnosti od vhodnih signalov lahko spremeni samo ob fronti urinega signala.
- CPE je zgrajena iz
  - kombinatoričnih in
  - sinhronskih (sekvenčnih) pomnilniških digitalnih vezij.
- Trenutno stanje vseh pomnilniških vezij predstavlja **stanje CPE.**

- Delovanje CPE je v vsakem trenutku odvisno od trenutnega stanja vhodov v CPE in od trenutnega stanja CPE.
- Število vseh možnih notranjih stanj CPE je odvisno od velikosti (zmogljivosti) CPE.
- Število bitov, s katerimi so predstavljena notranja stanja CPE, je od nekaj 10 pa do 10.000 ali tudi več.
- Digitalna vezja iz katerih je narejena CPE so danes običajno na enem čipu.

Fetch, Execute:

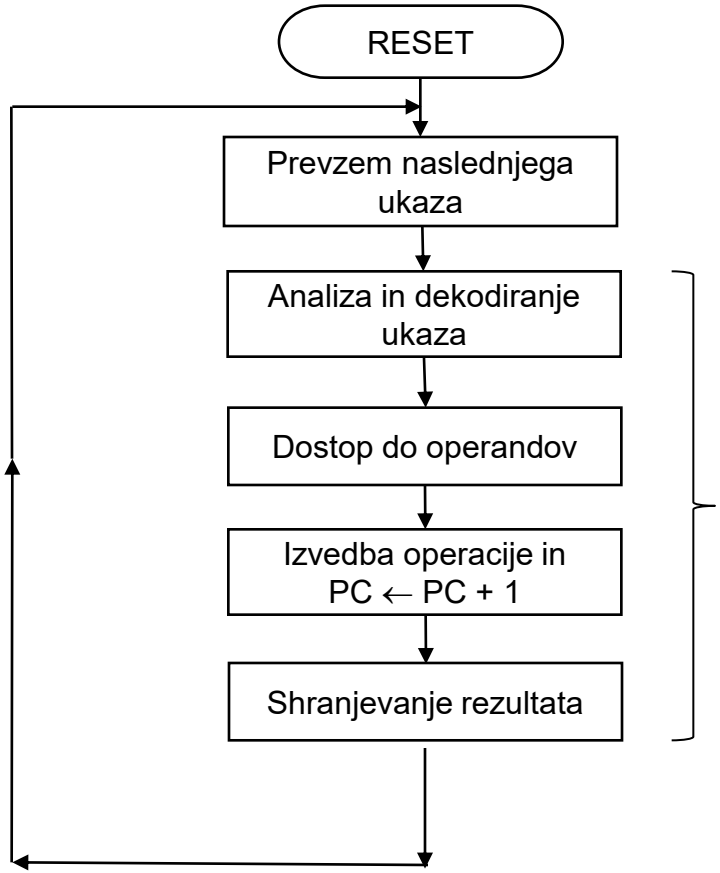
- Delovanje CPE v von Neumannovem računalniku smo opisali z dvema korakoma:
  - 1. Jemanje ukaza iz pomnilnika (ukazno-prevzemni cikel), naslov ukaza je v programskem števcu (PC)
  - 2. Izvrševanje prevzetega ukaza (izvršilni cikel),
- Vsako od teh dveh glavnih korakov lahko razdelimo še na bolj enostavne podoperacije („elementarne“ korake) ->

- Delovanje CPE v von Neumannovem računalniku smo opisali z dvema korakoma:
  - 1. Jemanje ukaza iz pomnilnika (ukazno-prevzemni cikel), naslov ukaza je v programskem števcu (PC)
  - 2. Izvrševanje prevzetega ukaza (izvršilni cikel), ki ga lahko razdelimo na več podoperacij:
    - Analiza (dekodiranje) ukaza
    - Prenos operandov v CPE (če niso že v registrih v CPE)
    - Izvedba z ukazom določene operacije
    - PC←PC+1 ali PC←ciljni naslov pri skočnih ukazih
    - Shranjevanje rezultata (če je potrebno)

Izvedba ukazov

Naslov prvega ukaza po vklopu (RESET) je določen z nekim pravilom.

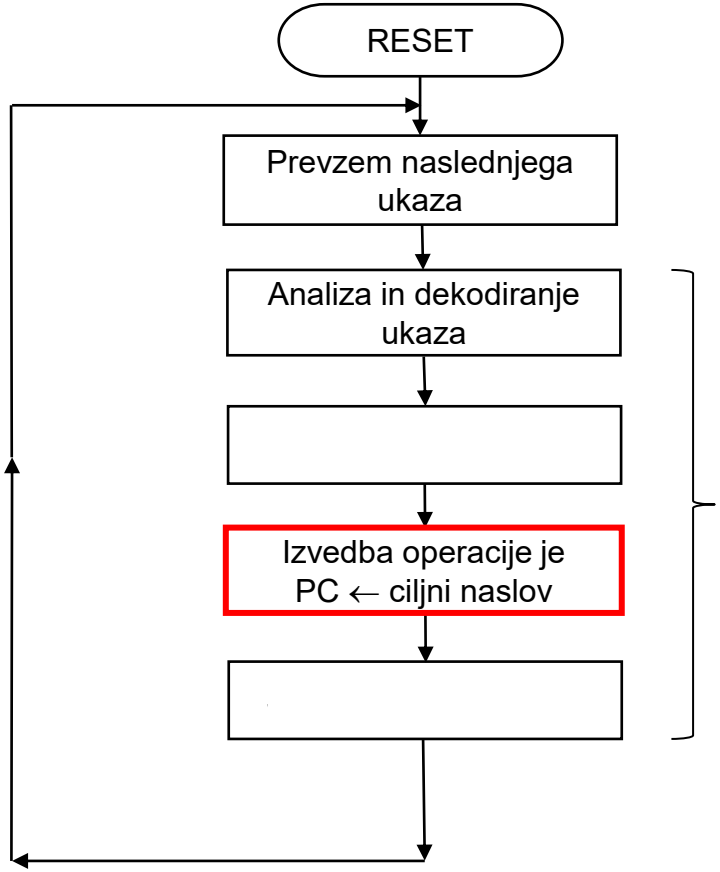
Po zaključku 2. koraka prične CPE zopet s 1. korakom, kar se ponavlja, dokler CPE deluje.



Vklop – začetno stanje

1. Ukazno-prevzemni cikel

2. Izvršilni cikel (AL ukaz)
npr: ADD R1,R2,R3

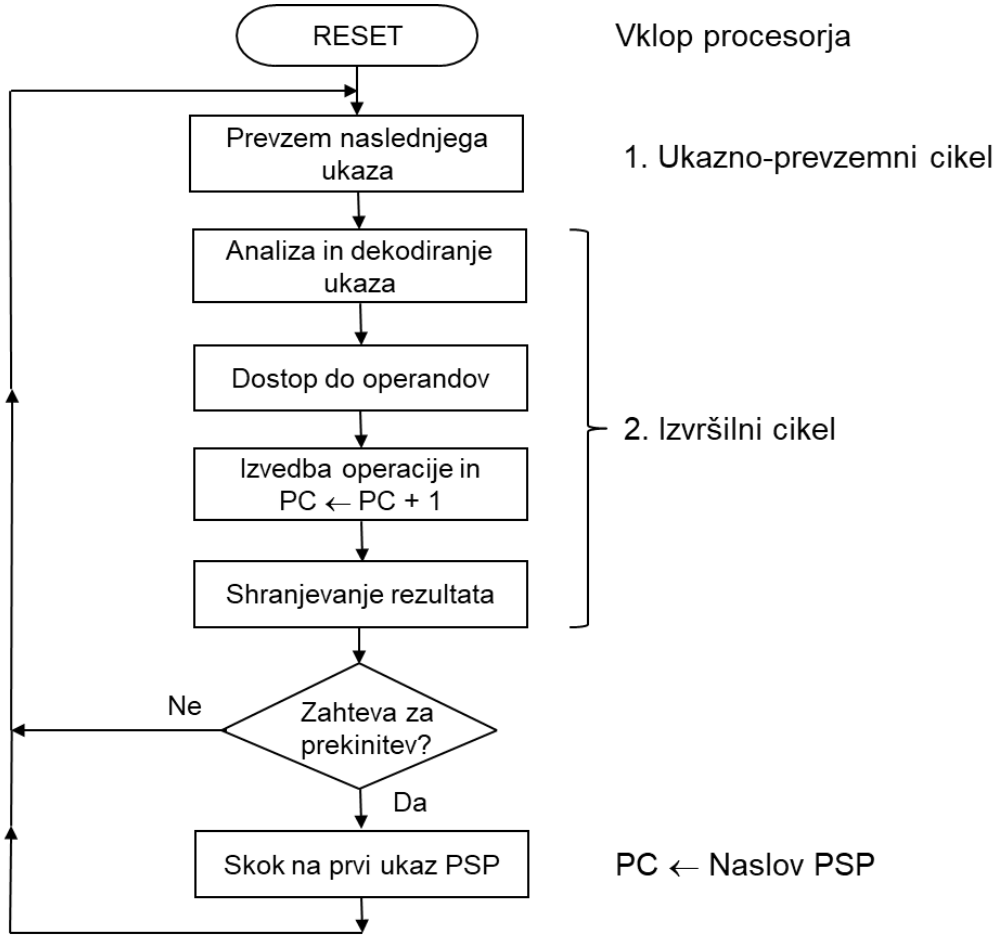


Vklop procesorja

1. Ukazno-prevzemni cikel

2. Izvršilni cikel (skočni ukaz)
   npr.: B OZNAKA

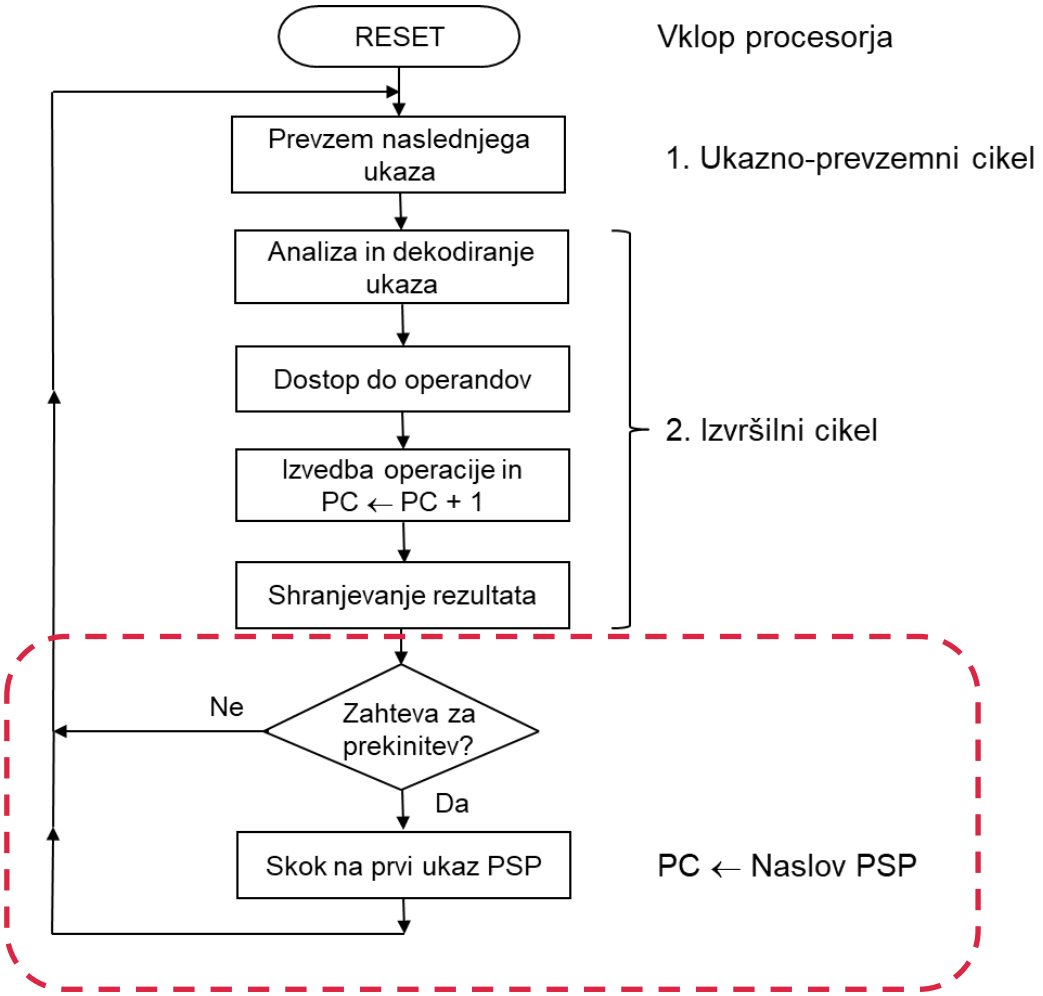
- Naslov prvega ukaza po vklopu (RESET) je določen s pravilom.
- Po zaključku 2. koraka prične CPE zopet s 1. korakom, kar se ponavlja, dokler CPE deluje.
- Izjema je, kadar pride do prekinitve ali pasti.
- Takrat se namesto prevzema naslednjega ukaza izvrši skok na ukaz, katerega naslov je določen z načinom delovanja prekinitev (PSP).



Prekinitve ali pasti:

- izredni dogodki
- transparentnost

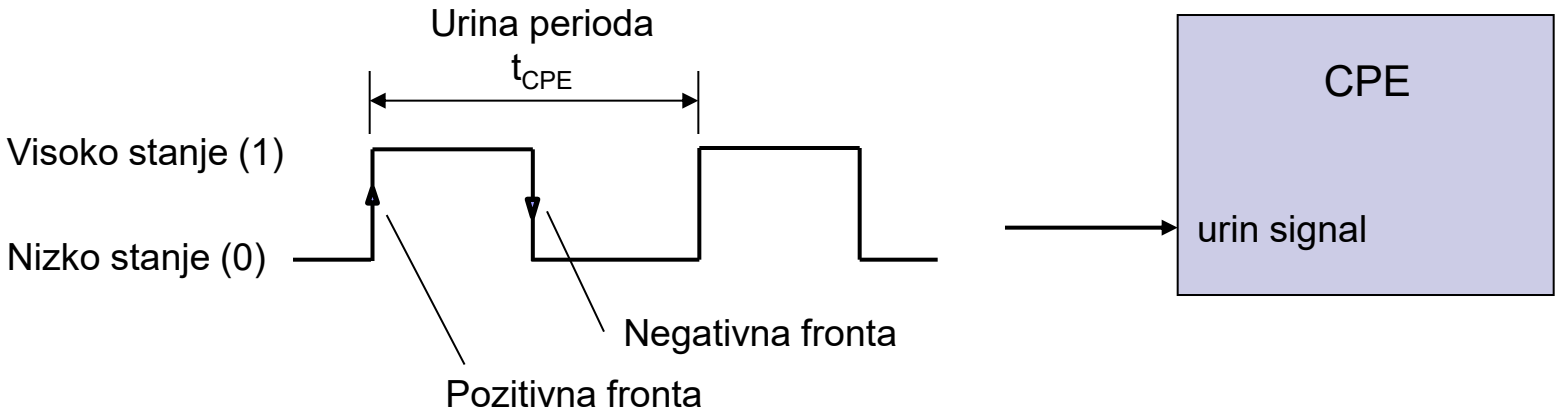
- takrat se namesto prevzema naslednjega ukaza izvrši skok na ukaz, katerega naslov je določen z načinom delovanja prekinitev (PSP).



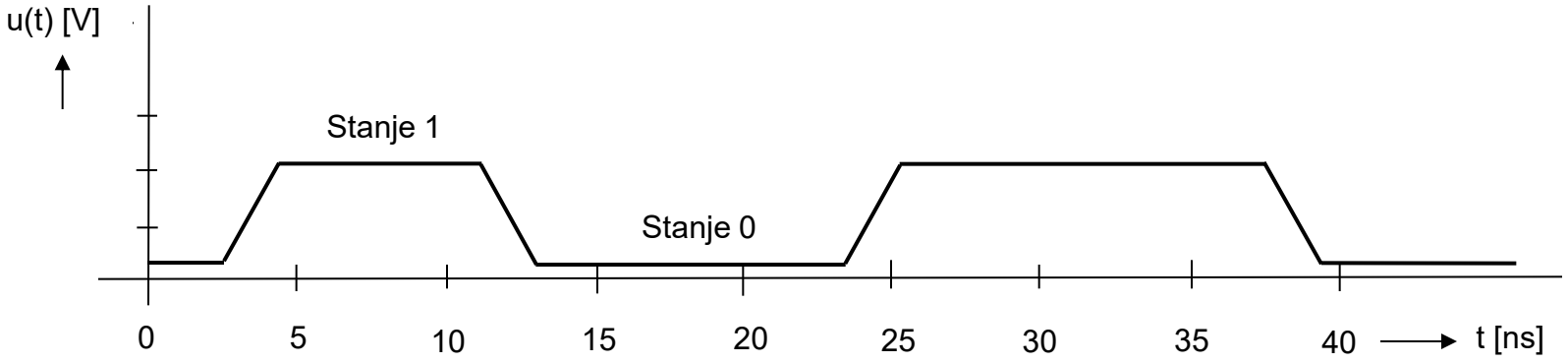
Urin signal

- Vsak od teh korakov je sestavljen iz bolj elementarnih korakov in realizacija CPE je realizacija teh elementarnih korakov.

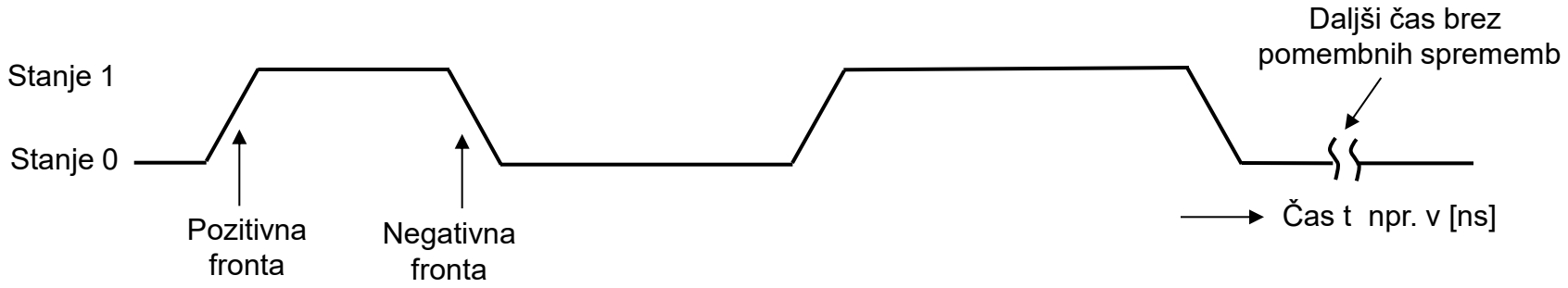
- Vsak elementarni korak se opravi v eni ali več periodah urinega signala – CPE ure.



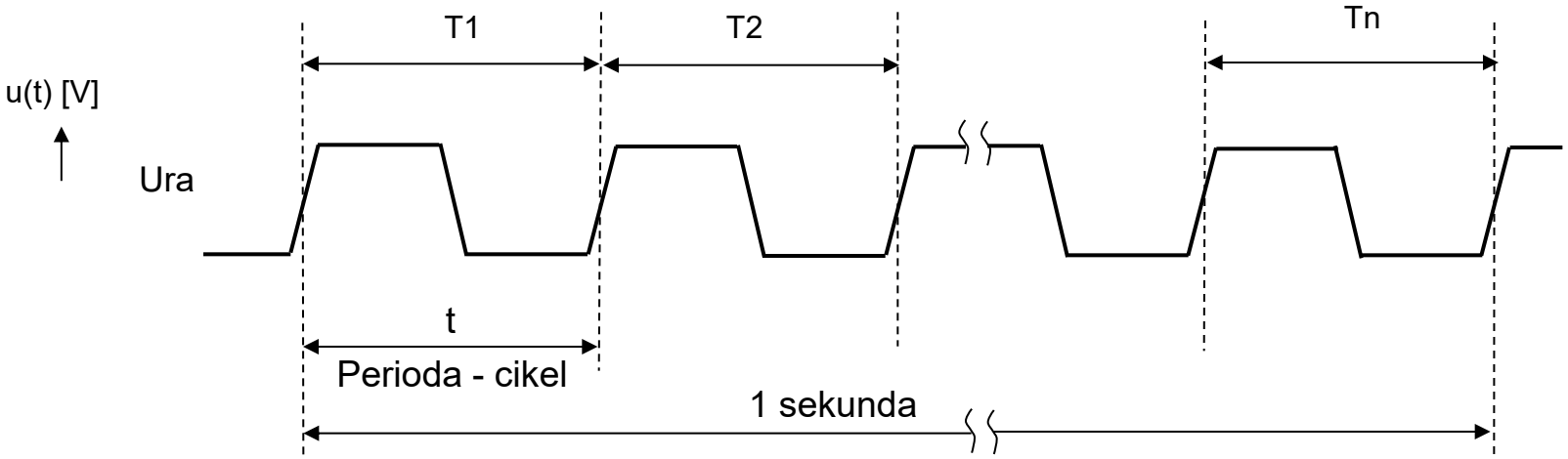
Poljuben (neperiodičen) digitalni električni signal



Poljuben (neperiodičen) digitalni signal – logična predstavitev



Urin signal - periodičen pravokoten signal



Pri f = 1,25 GHz je v 1 sekundi 1.250.000.000 period

Frekvenca periodičnega signala f = število period (ciklov) v 1 sekundi

Enota za frekvenco je Hertz [Hz] : 1 Hz = 1 [perioda/s] = 1[1/s] = 1[s^{-1}]

Čas trajanja ene periode t = 1 / f

$$f = 1{,}25[GHz] \Rightarrow t = \frac{1}{f} = \frac{1}{1{,}25 * 10^{9}[1/s]} = \frac{1}{1{,}25} * 10^{-9}[s] = 0{,}8 * 10^{-9}[s] = 0{,}8[ns]$$

- Stanje CPE se, tako kot stanja vseh sinhronskih digitalnih vezij, spreminja samo ob fronti urinega signala (prehodu urinega signala iz enega v drugo stanje).

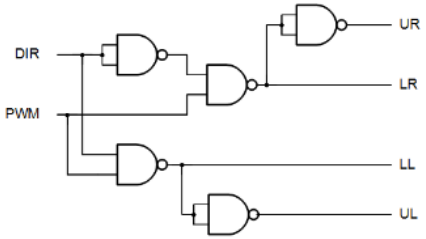
- Fronto ob kateri se dogajajo spremembe v CPE imenujemo **aktivna fronta**.

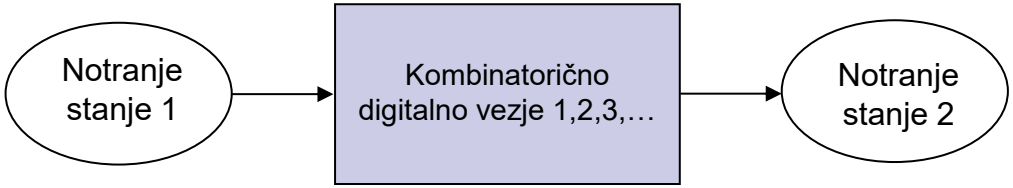
- CPE je lahko narejena tudi tako, da spreminja stanje ob pozitivni in negativni fronti, to pomeni, da sta aktivni obe fronti. V eni urini periodi se tako lahko izvršita dve spremembi stanja CPE.

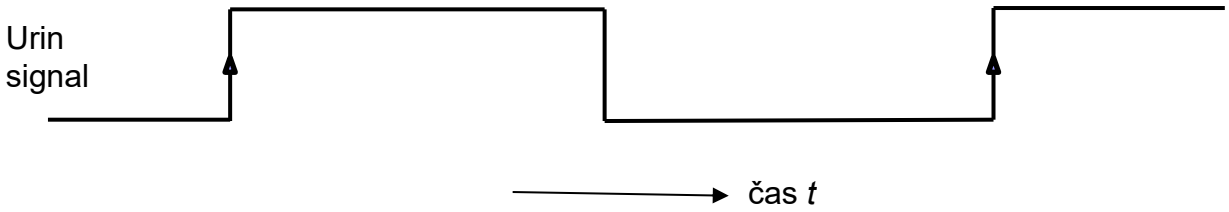
Zakaj je urin signal sploh potreben ? 2 vidika ->

Urin signal -> sinhronizacija različno hitrih komb. vezij v računalniku

- V sinhronskem pomnilniškem digitalnem sistemu urin signal (običajno fronta) določa trenutek spremembe notranjega stanja v pomnilniškem digitalnem vezju.
- Ko vhodni signali v pomnilniško vezje postanejo stabilni, se ob urini fronti lahko spremeni notranje stanje pomnilniškega vezja.

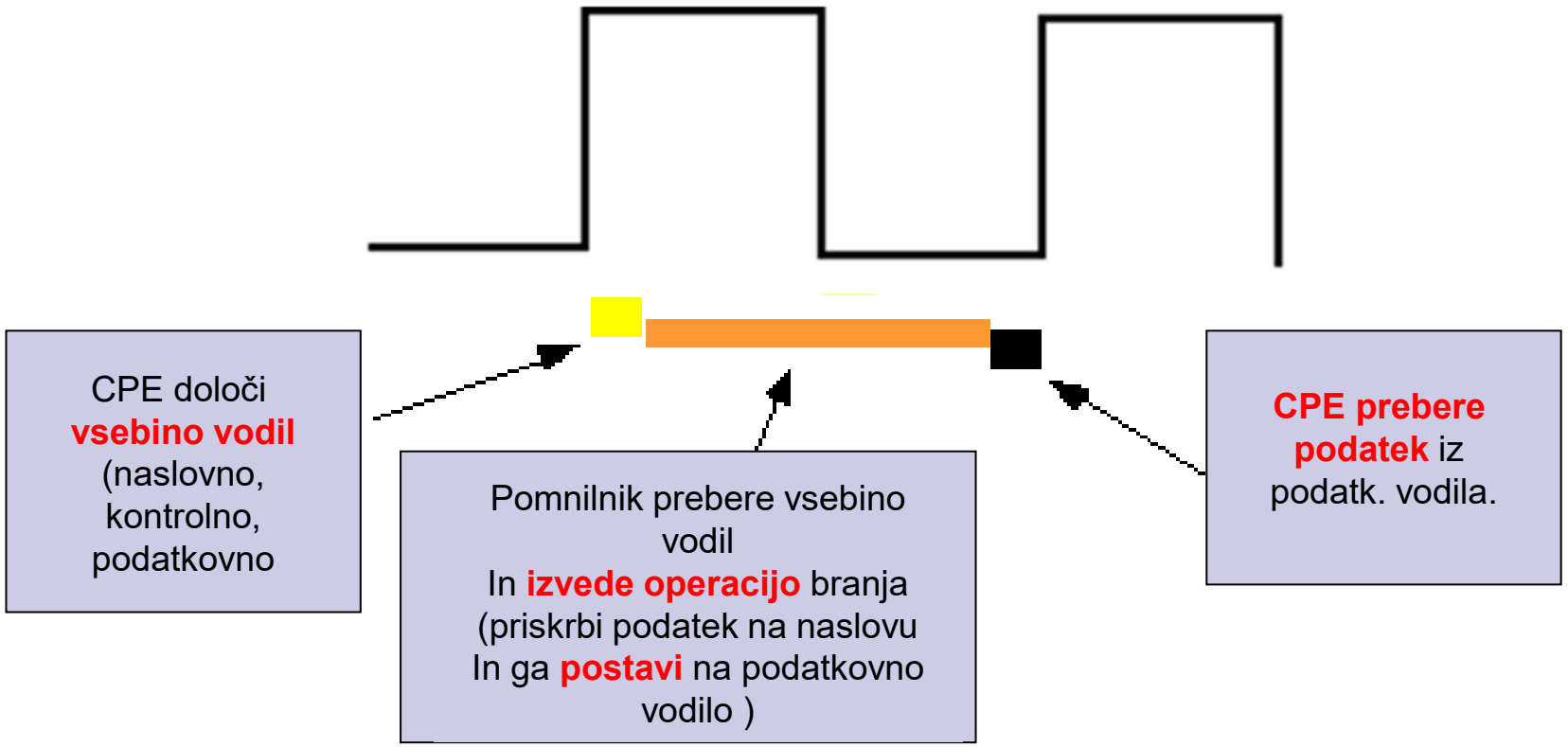






Urin signal -> sinhronizacija različno hitrih operacij v računalniku

- Primer dostopa do pomnilnika v 1 urinem ciklu (branje) :



- Stanje CPE se spreminja ob frontah notranje ure. Krajša urina perioda (višja frekvenca) pomeni hitrejše delovanje CPE.
- Krajšanje urine periode (višanje frekvence) je pogojeno s hitrostjo uporabljenih digitalnih vezij in številom vezij (dolžino povezav) skozi katera potuje signal.
- Najkrajše trajanje elementarnega koraka v CPE je ena urina perioda (ali tudi pol periode, če sta aktivni obe fronti, za kar pa je potrebno bolj kompliciran logično vezje).
- Prevzemni in izvršilni cikel trajata vedno celo število urinih period.
- Število urinih period za izvedbo ukaza se med ukazi lahko precej razlikuje.

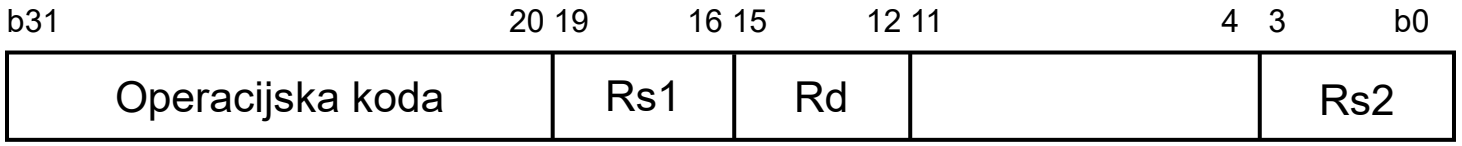
6.2 ARM9 CPE – povzetek lastnosti

Veliko več informacij povemo na laboratorijskih vajah

- RISC arhitektura (nekaj izjem)
- 3-operandni registrsko-registrski (load/store) računalnik
  - Dostop do pomnilniških operandov je samo z ukazoma LOAD in STORE
- 32-bitni računalnik (FRI-SMS, ARM9, arhitektura ARMv5)
  - 32-bitni pomnilniški naslov
  - 32-bitno podatkovno vodilo
  - 32-bitni registri
  - 32-bitna ALE
- 16 splošno namenskih 32-bitnih registrov
- Dolžine pomnilniških operandov 8, 16 in 32 bitov
- Predznačena števila so predstavljena v dvojiškem komplementu
- Realna števila po standardu IEEE-754 (v primeru FP-enote)

- Sestavljeni pomnilniški operandi so shranjeni po pravilu tankega konca.
- Ukazi in operandi morajo biti v pomnilniku poravnani (shranjeni na naravnih naslovih).
- Vsi ukazi so dolgi 32 bitov (4 bajte).
- ARM uporablja vse tri načine naslavljanja:
  - Takojšnje — *ADD r1,r1,#1*
  - Neposredno (registrsko) — *ADD r1,r1,r2*
  - Posredno (registrsko)-LOAD/STORE — *LDR r1,[r0]*

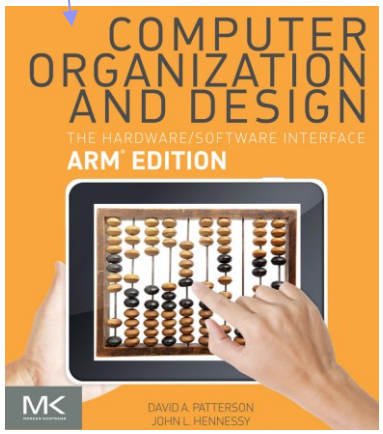
- Ukazi za pogojne skoke uporabljajo PC-relativno naslavljanje.
- Primer formata ALE ukaza:

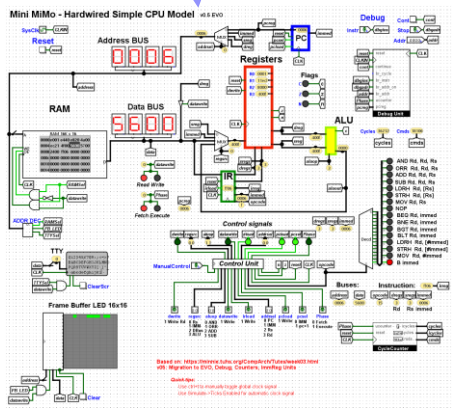


6.3 Zgradba CPE (primera ARM CPE LEGv8 in [Mini]MiMo)

6.3.1 Podatkovna enota ->

- ALE ->
- Programsko dostopni registri ->



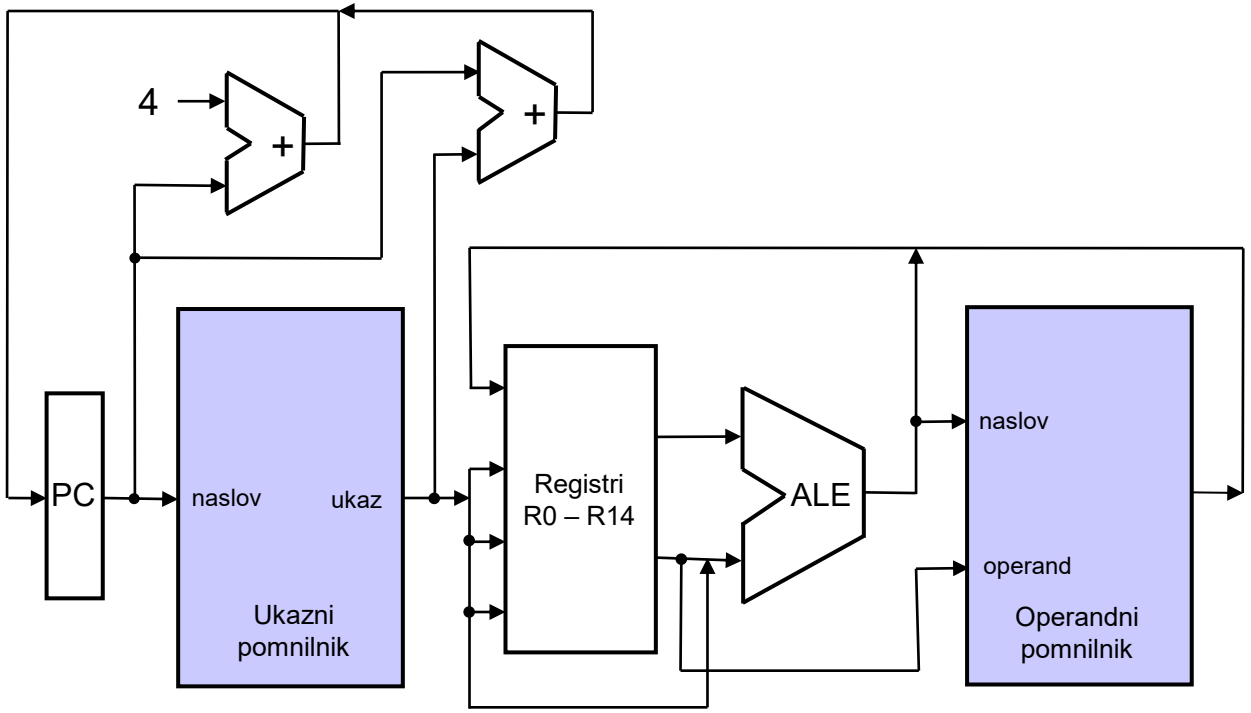


6.3.2 Kontrolna enota ->

- Realizacija
  - mikroprogramska (SW) ali
  - trdožičena (HW)

6.3.1 Podatkovna enota

Poenostavljena zgradba CPE s podatkovnimi potmi in ukaznim ter operandnim pomnilnikom



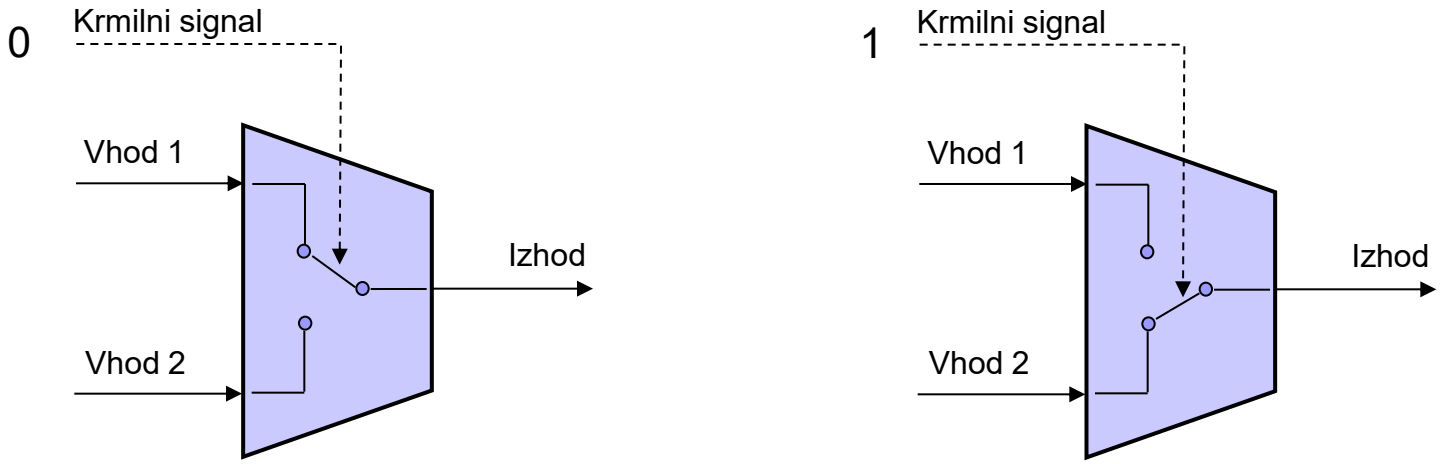
Vse podatkovne poti so M-bitne, puščice nakazujejo smer prenosa

Poenostavljena različica ARMv8 (Vir: [Patt], pogl. 4)

MUX – Multiplekser

- MUX – Multiplekser: je digitalno vezje, ki iz več vhodnih signalov izbere enega in ga posreduje na izhod.

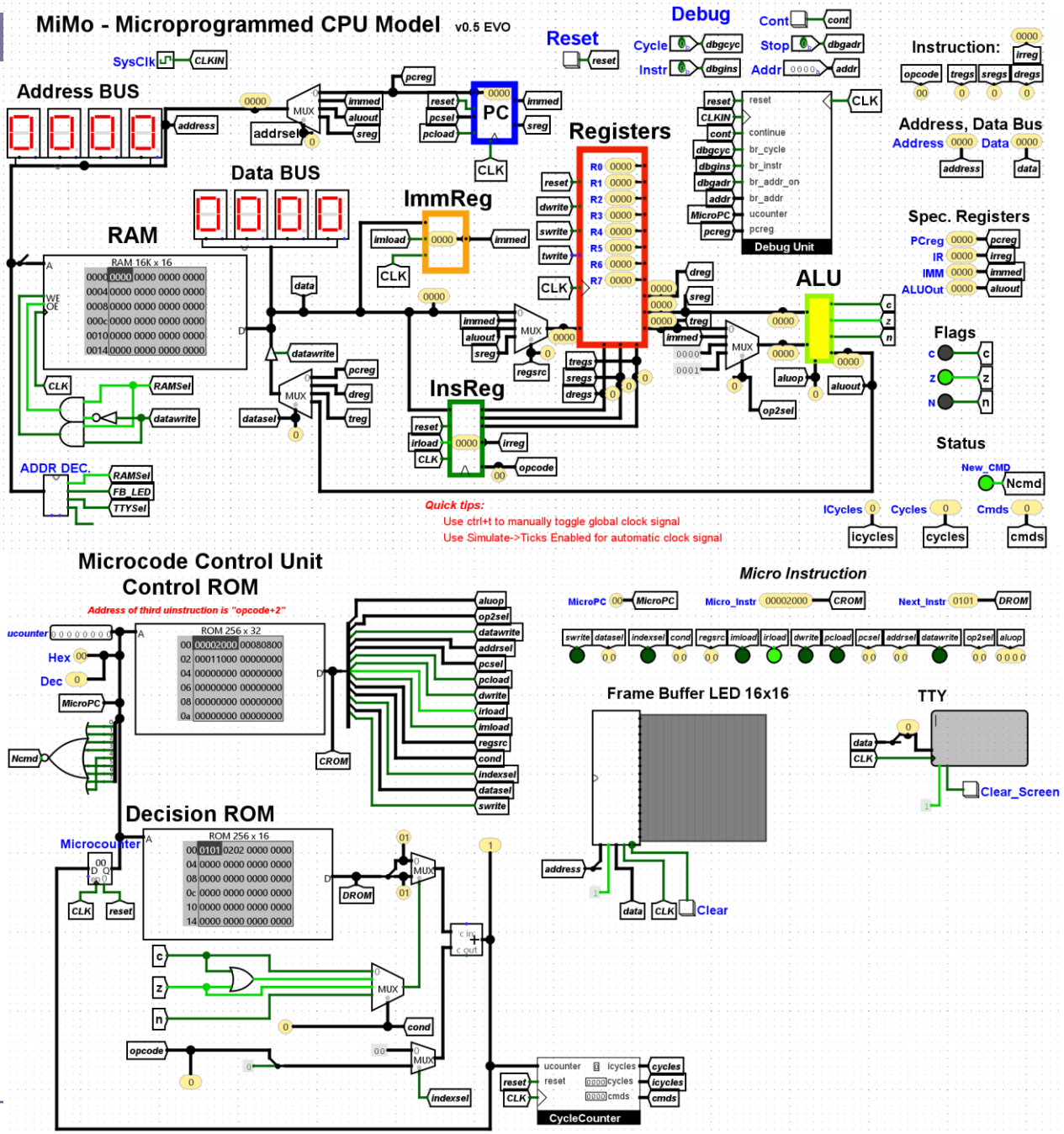
- Izbiro vhodnega signala določa krmilni signal.



Primer CPE: MiMo

Model CPE realiziran z logičnimi vrati v Logisimu

MiMo – Mikroprogramiran Model CPE (predmet OR VSP)



https://github.com/LAPSyLAB/MiMo_Student_Release

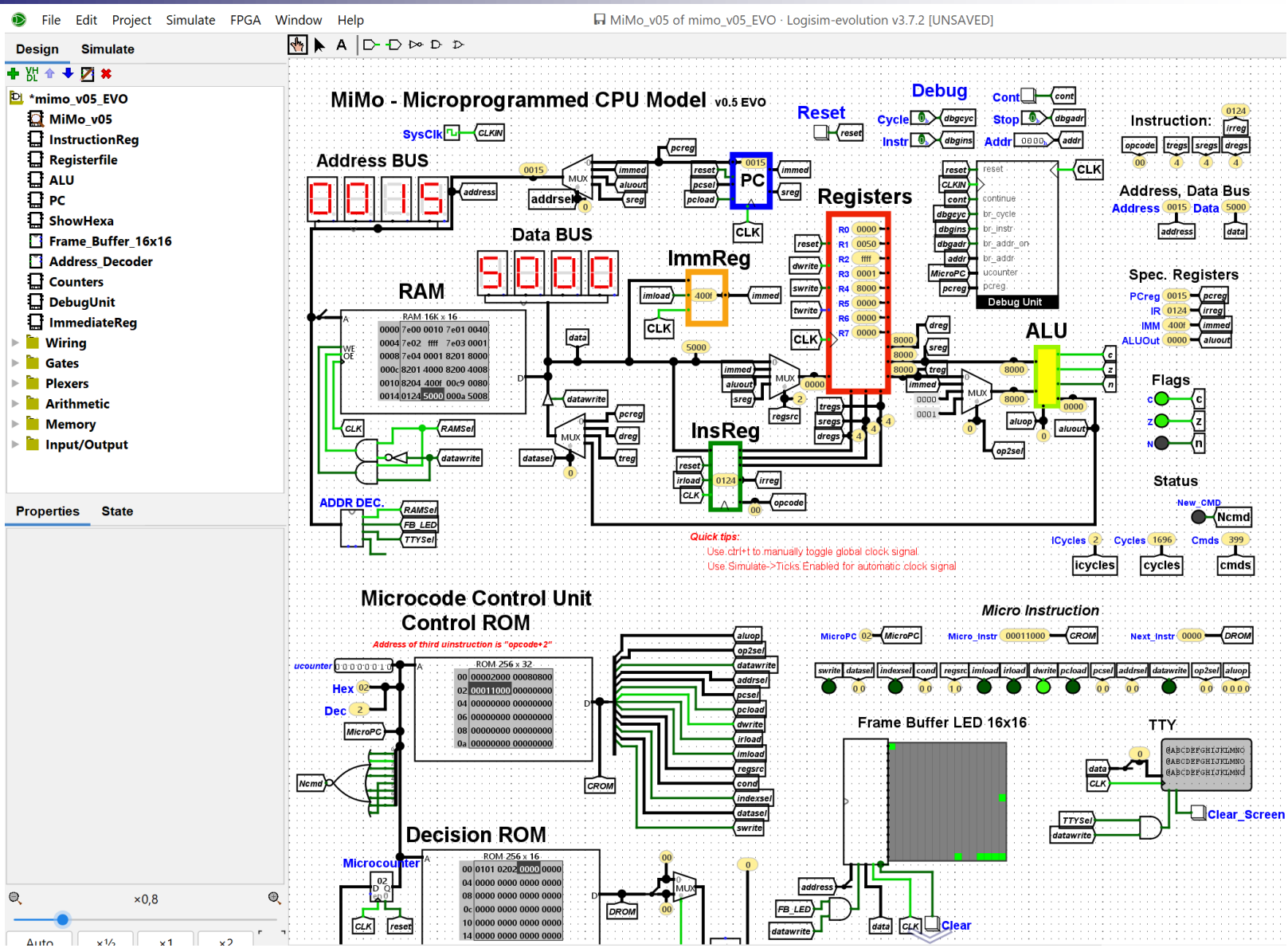
Mimo Video

Primer CPE: MiMo

Model CPE realiziran z logičnimi vrati v Logisimu

MiMo – Mikro programiran Model CPE

Video



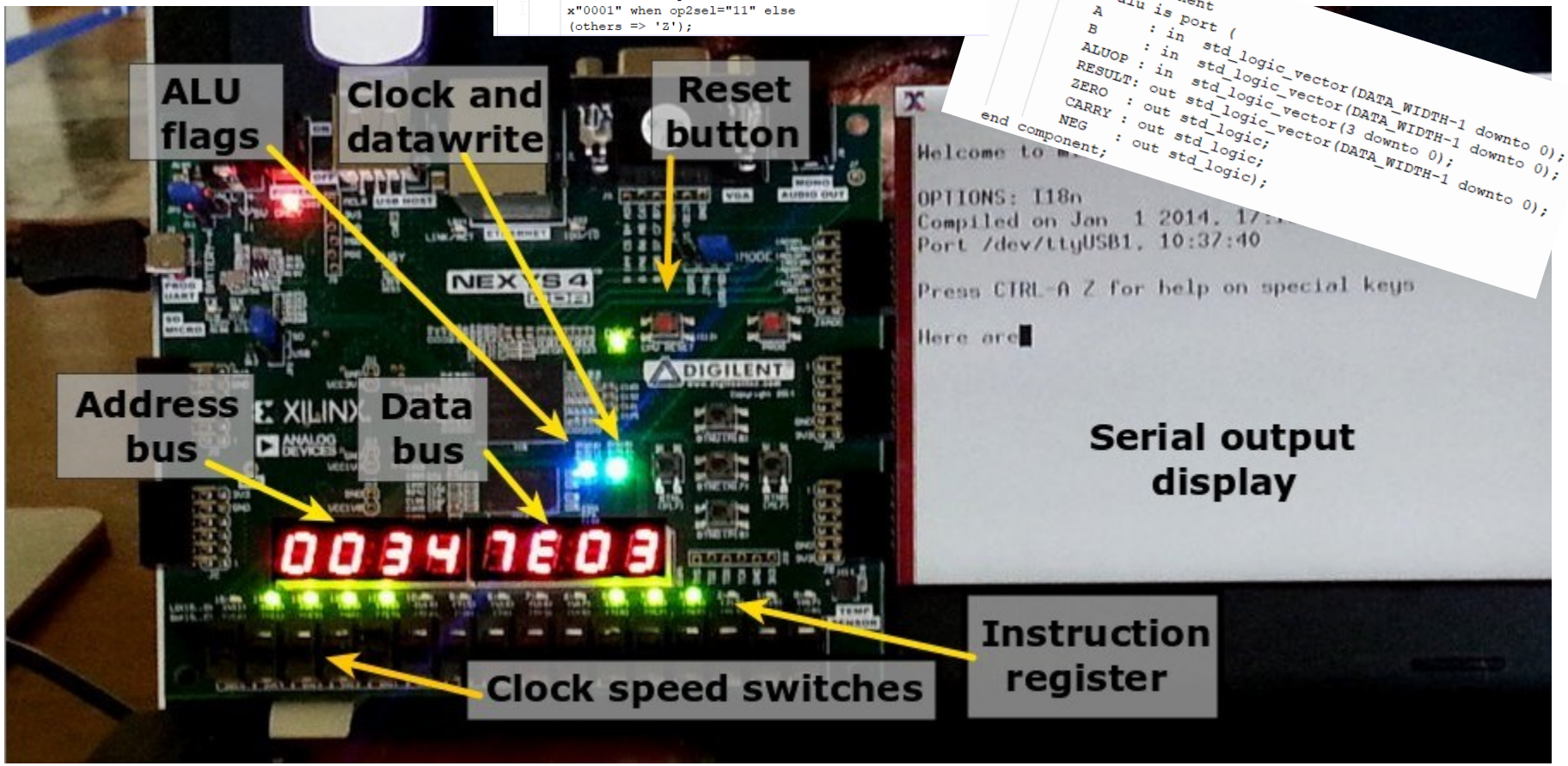
MiMo – Mikro-programiran Model CPE

FPGA realizacija

```
-- Multiplexor into the second ALU input, and first ALU input
alu_a <= sreg;
alu_b <= treg when op2sel="00" else
         immed_out when op2sel="01" else
         x"0000" when op2sel="10" else
         x"0001" when op2sel="11" else
         (others => 'Z');
```

```
VHDL:
   -- The ALU component
   component alu is port (
         A      : in  std_logic_vector(DATA_WIDTH-1 downto 0);
         B      : in  std_logic_vector(DATA_WIDTH-1 downto 0);
         ALUOP  : in  std_logic_vector(3 downto 0);
         RESULT: out std_logic_vector(DATA_WIDTH-1 downto 0);
         ZERO   : out std_logic;
         CARRY  : out std_logic;
         NEG    : out std_logic);
   end component;
```

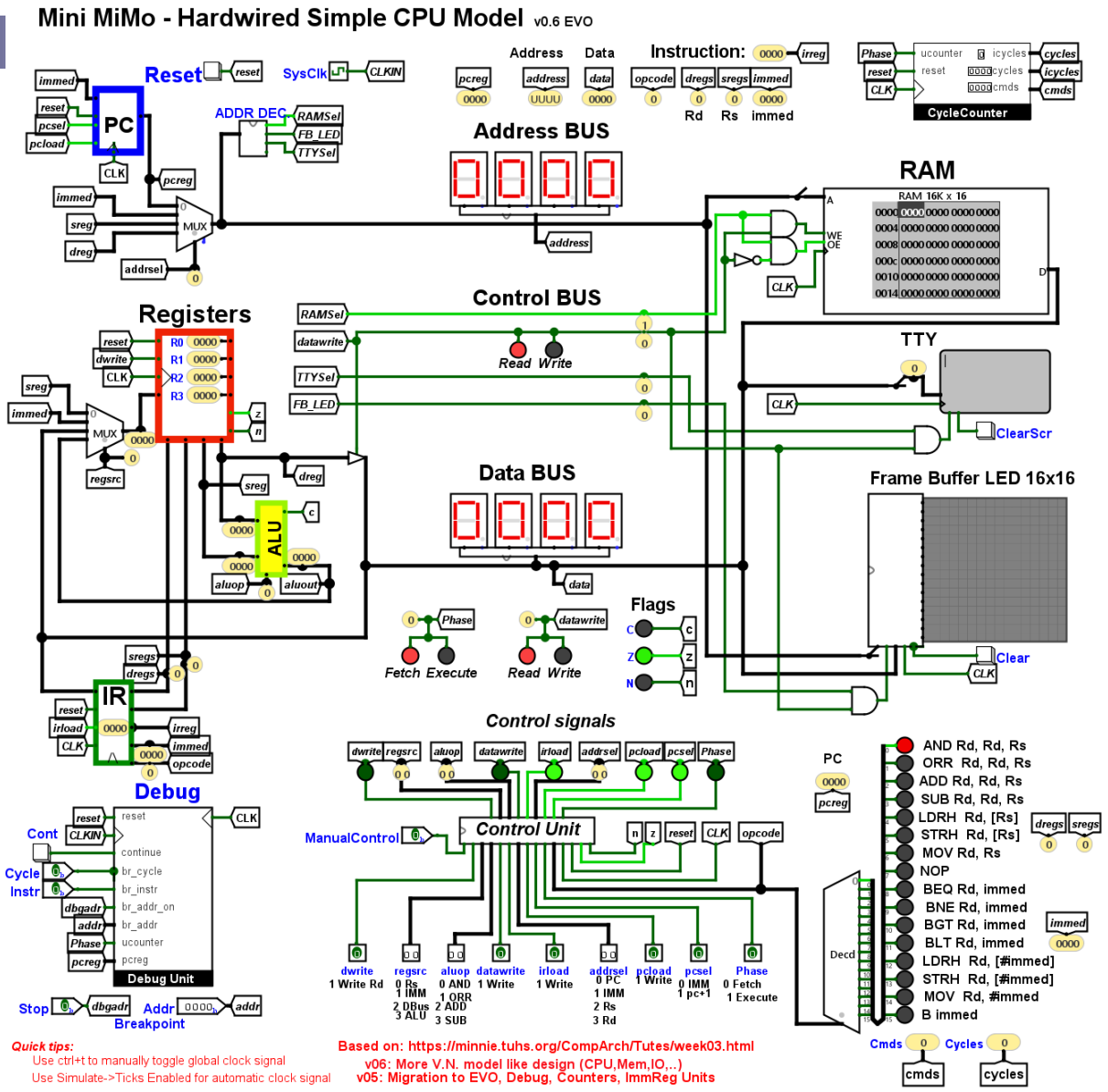
```
   -- The ALU
alunit: alu port map (
    A => alu_a,
    B => alu_b,
    ALUOP => aluop,
    RESULT=> aluout,
    ZERO => zero,
    CARRY => carry,
    NEG => neg
  );
```



Primer CPE: Mini MiMo (predmet RA VSP)

Model CPE realiziran z logičnimi vrati v Logisimu

Mini MiMo – Enostaven trdoožičen Model CPE (16 ukazov, zbirnik v Excelu, …)

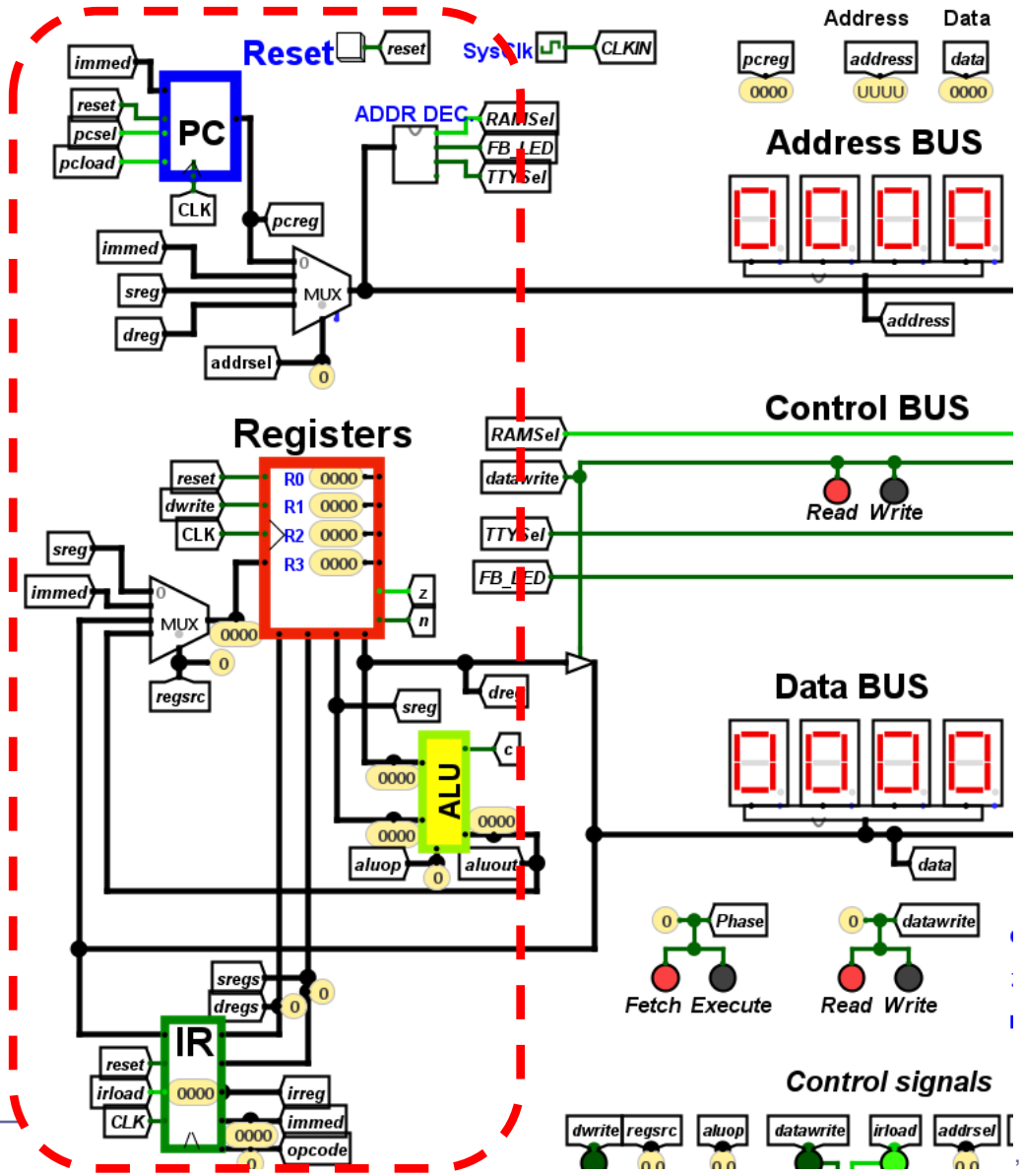


https://github.com/LAPSyLAB/RALab-STM32H7/tree/main/MiniMiMo_HW_CPE_Model

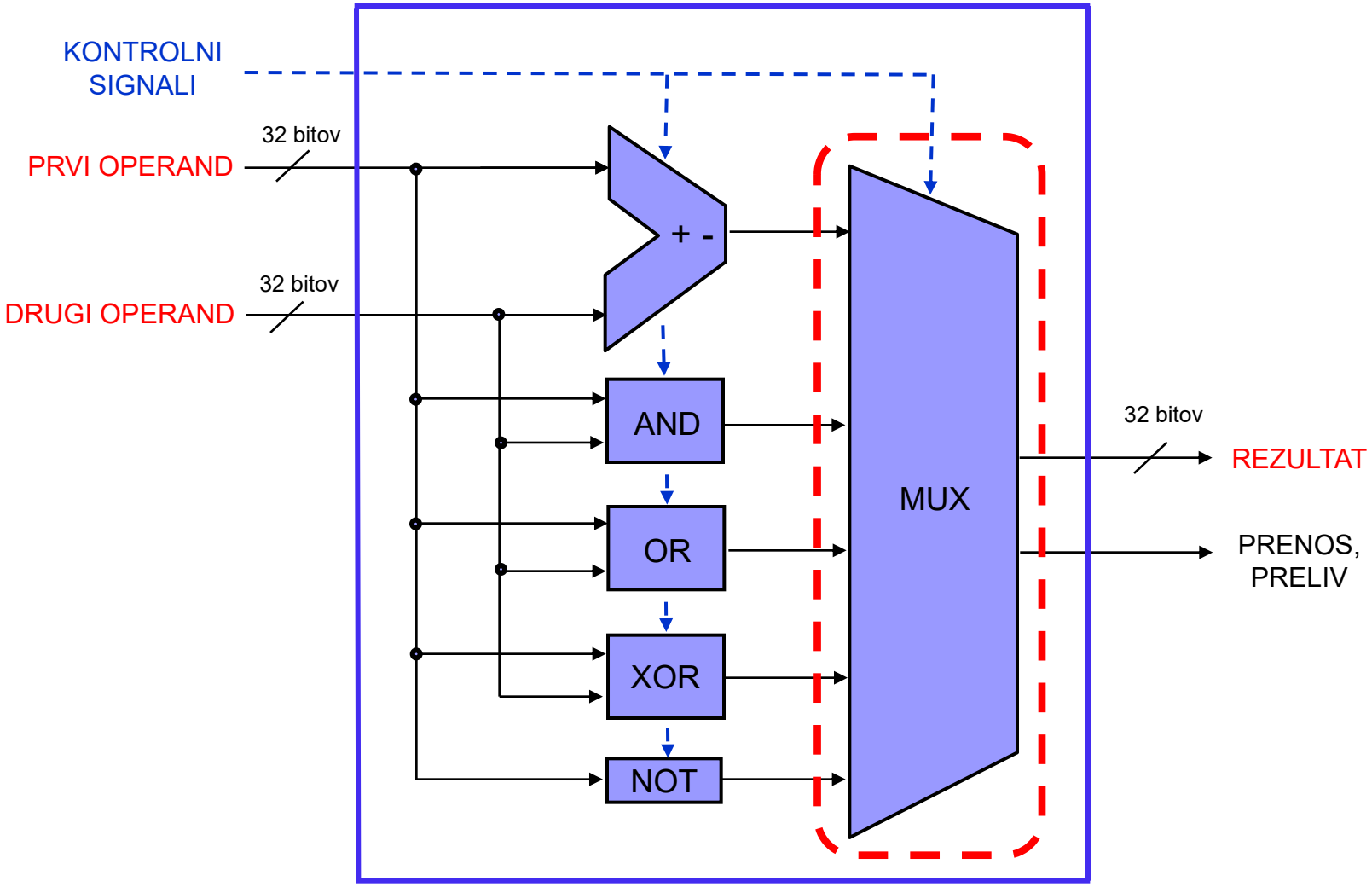
Mini MiMo podatkovna enota

6.3.1 Podatkovna enota

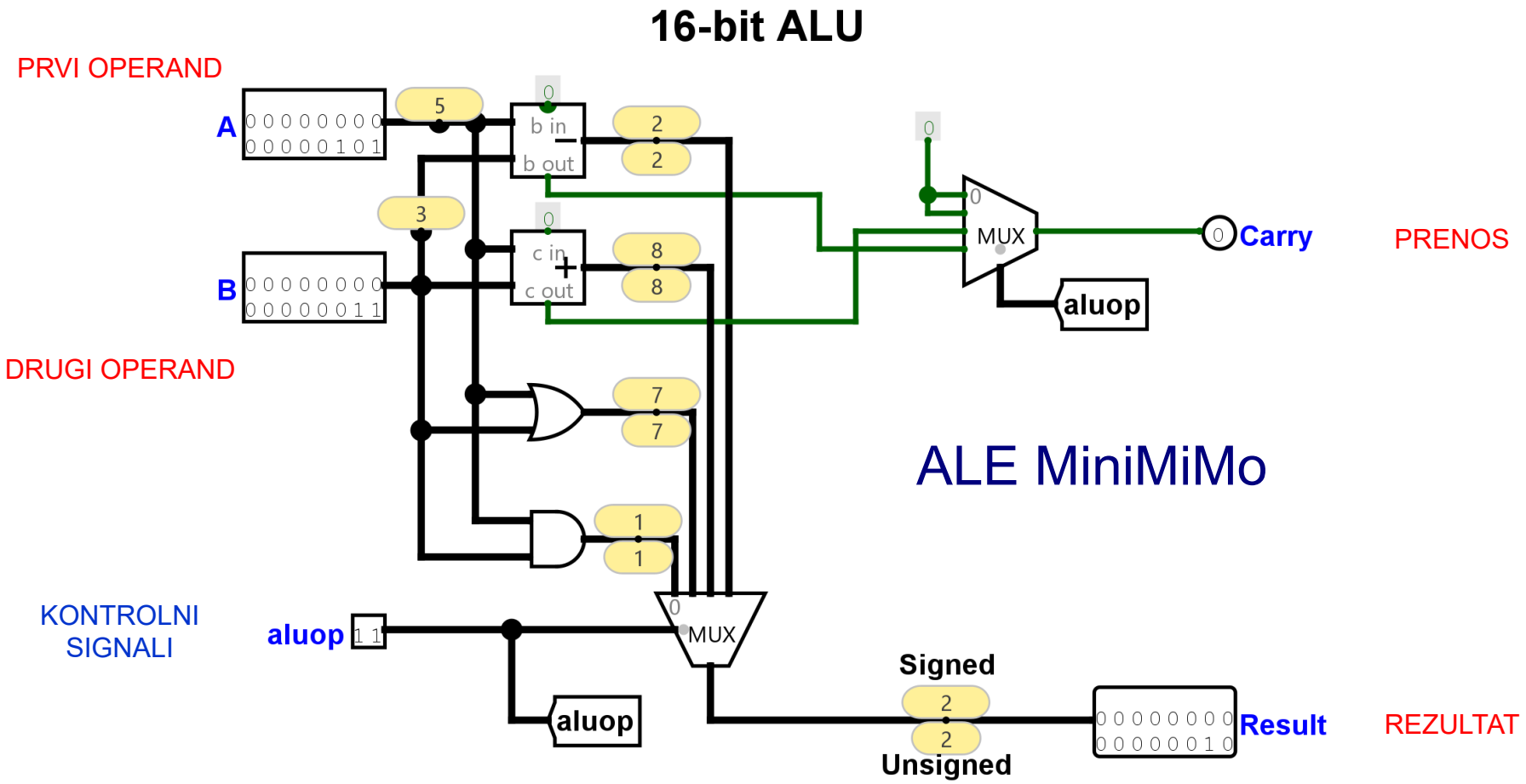
Debelejše povezave so več-bitne



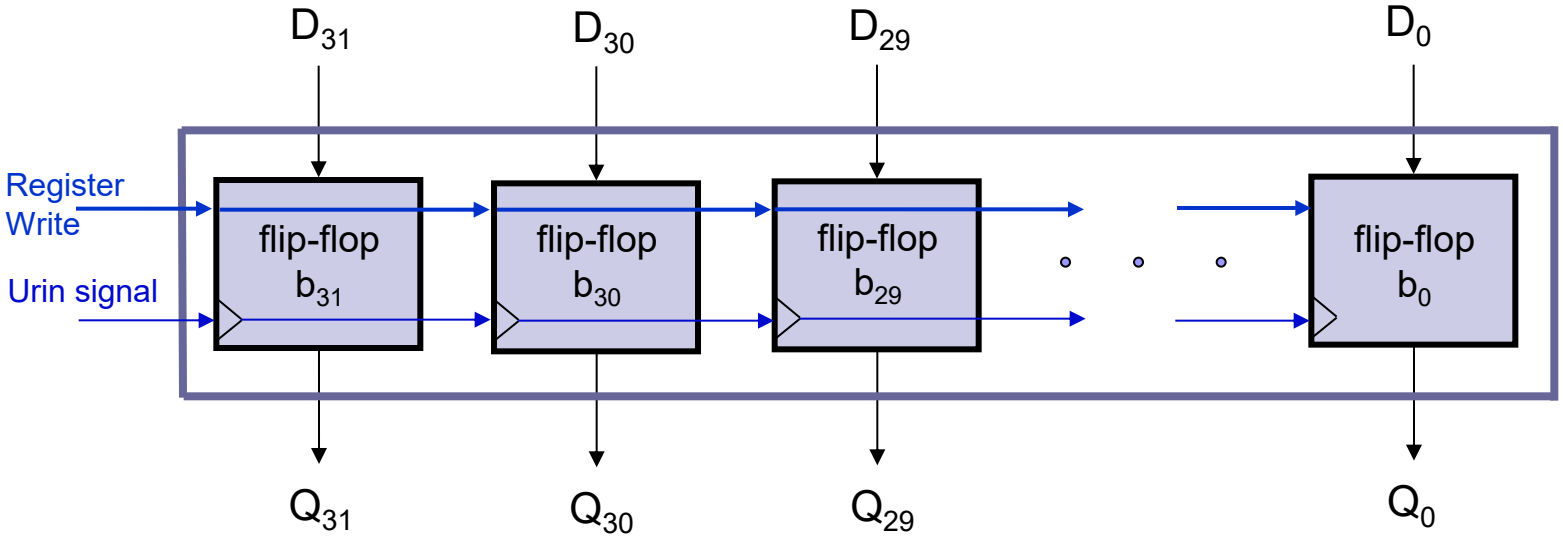
ALE – zgradba s podatkovnimi in kontrolnimi signali

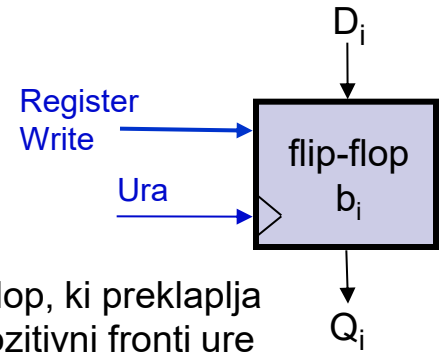


ALE – podatkovni in kontrolni signali primer Mini MiMo CPE



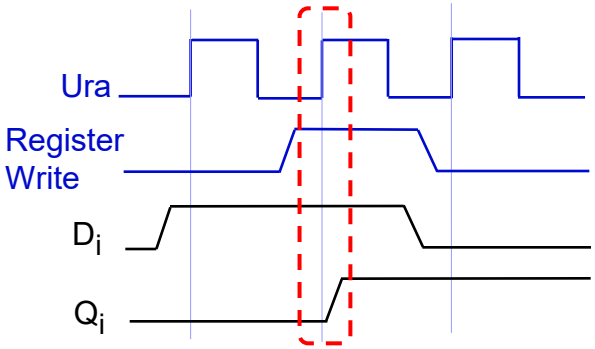
32-bitni register





Flip-flop, ki preklaplja ob pozitivni fronti ure

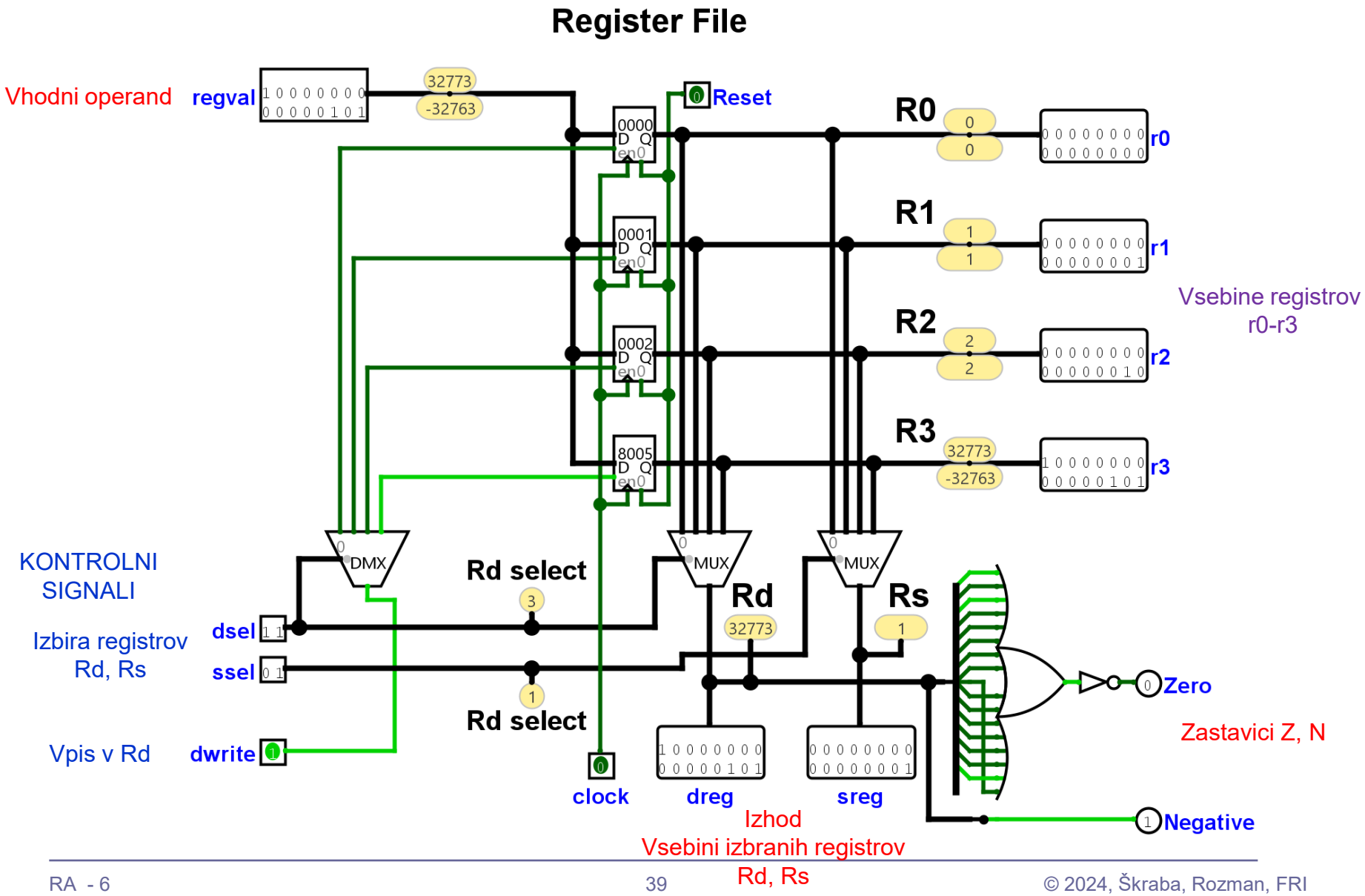
Časovni diagram



Pravilnostna tabela

Ura	Reg.W	D_i	Q_i
↑	0	0	Q
↑	0	1	Q
↑	1	0	0
↑	1	1	1

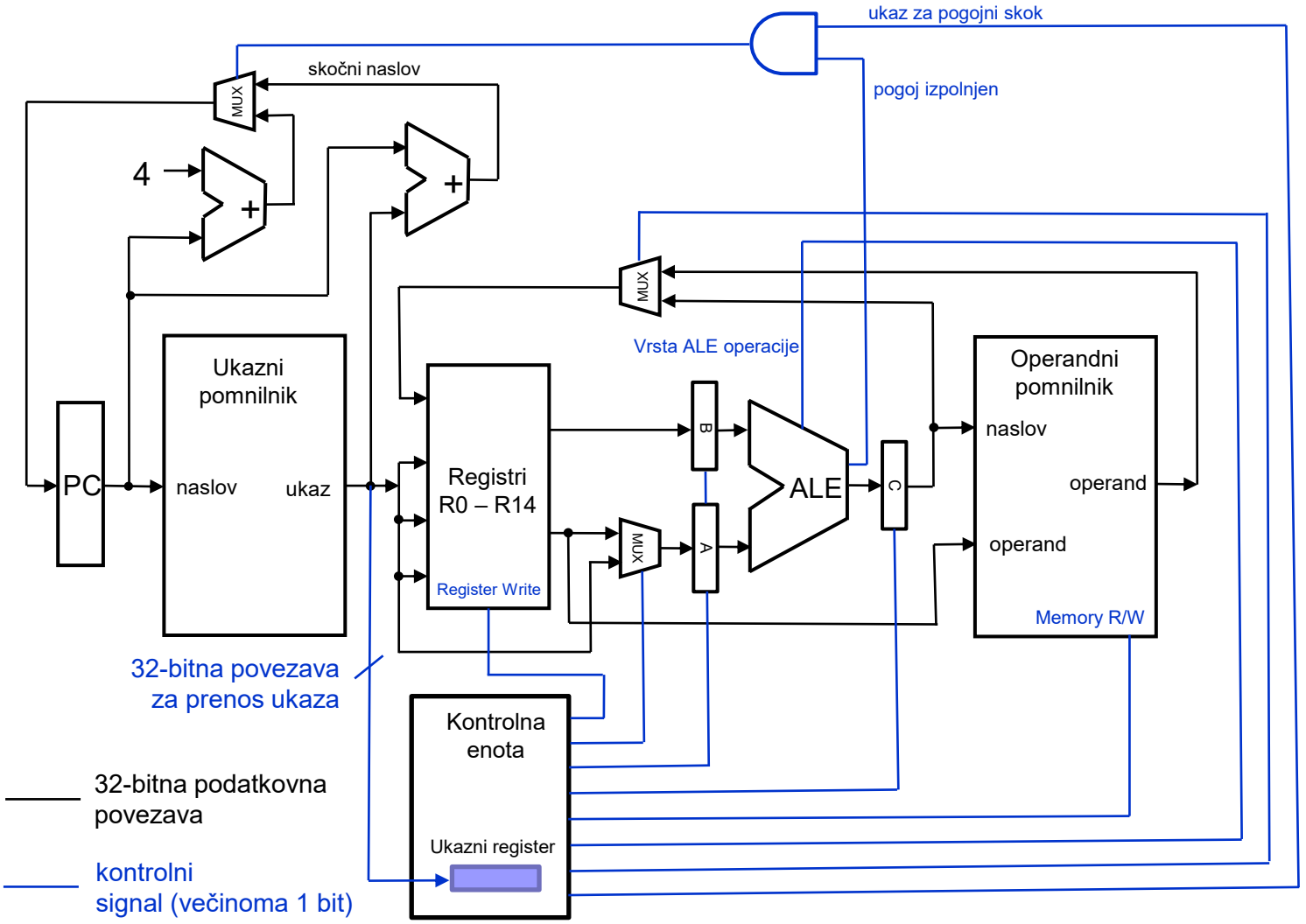
Registrska enota - Mini MiMo CPE



6.3.2 Kontrolna enota

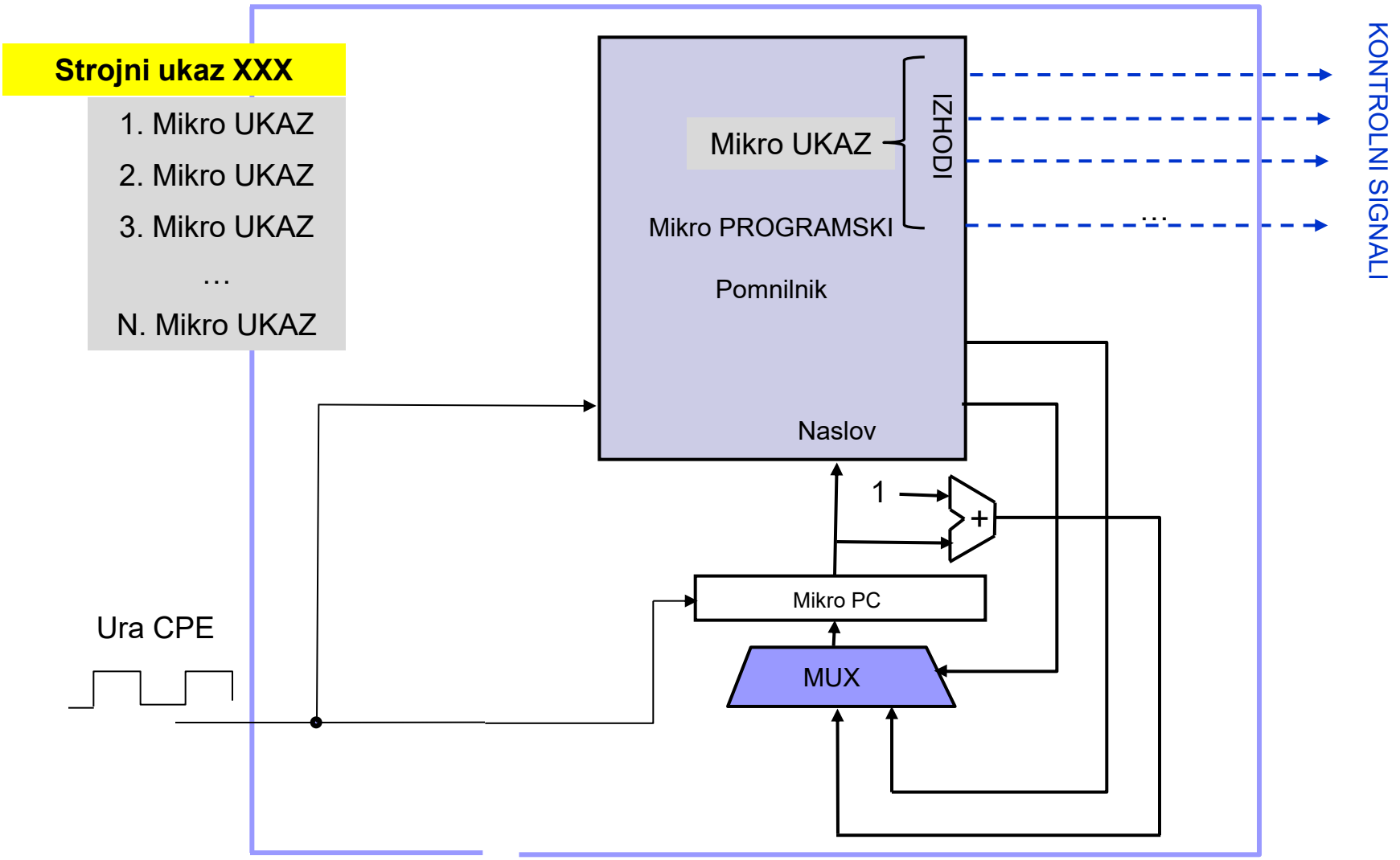
- Digitalno vezje (pomnilniško + kombinatorično), ki na osnovi ukaza v ukaznem registru tvori kontrolne signale.
- Kontrolni signali sprožajo elementarne korake v podatkovni enoti in s tem delne izvedbe ukaza.
- IR register = 32-bitni ukazni register v katerega se v ukazno-prevzemnem ciklu prenese strojni ukaz iz pomnilnika.
  - IR … „Instruction Register“
- 2 možna načina realizacije KE:
  - Mikroprogramska (SW: enostavna, počasnejša)
  - Trdoožičena (HW: zapletena, hitrejša)

CPE s podatkovno in kontrolno enoto ter kontrolnimi signali



Poenostavljena različica ARMv8 (Vir: [Patt], pogl. 4)

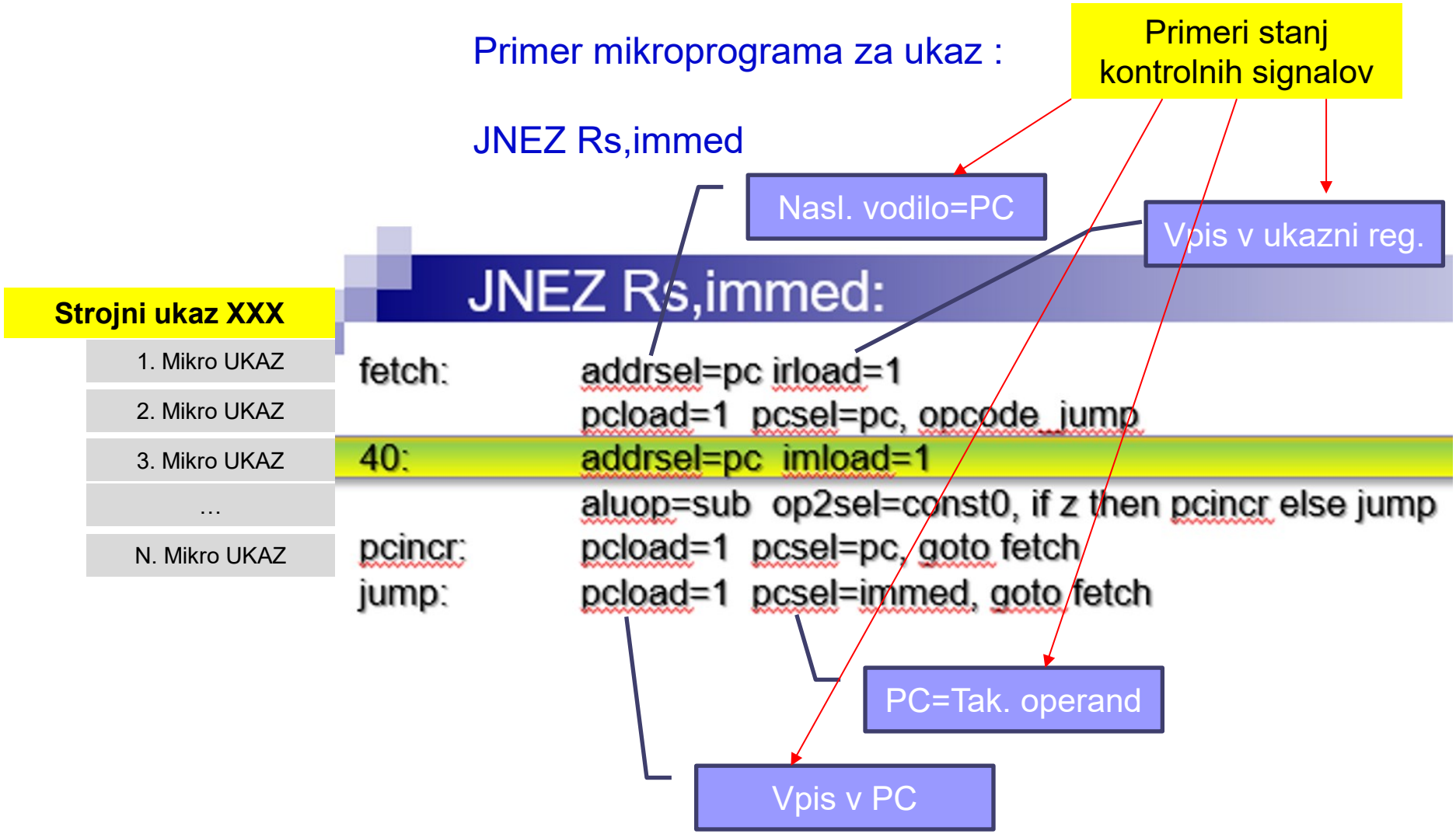
Kontrolna enota (mikroprogramska realizacija – npr. MiMo model)



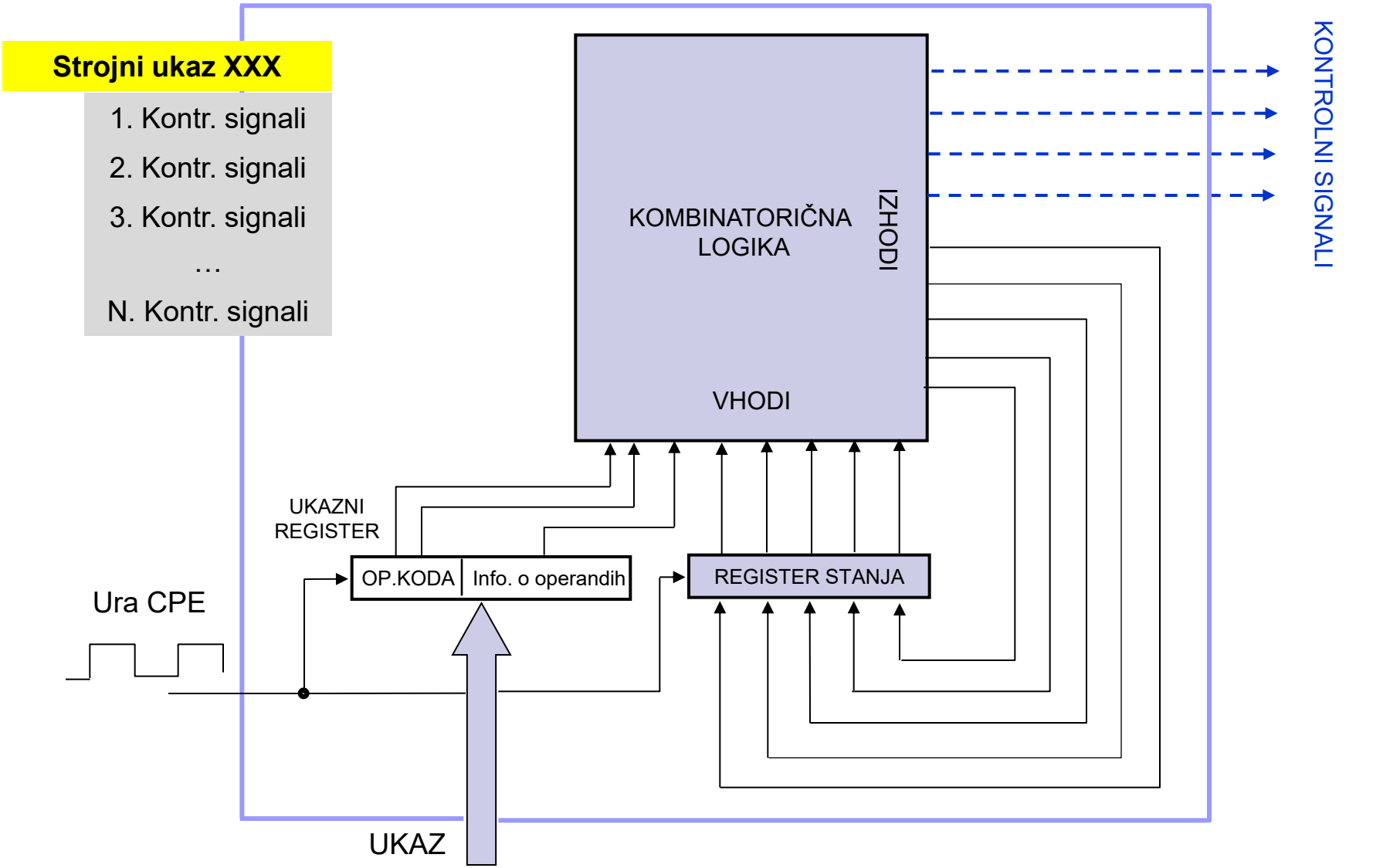
Kontrolna enota (primer mikroprogramske realizacije - MiMo model)

Primer mikroprograma za ukaz :

JNEZ Rs,immed



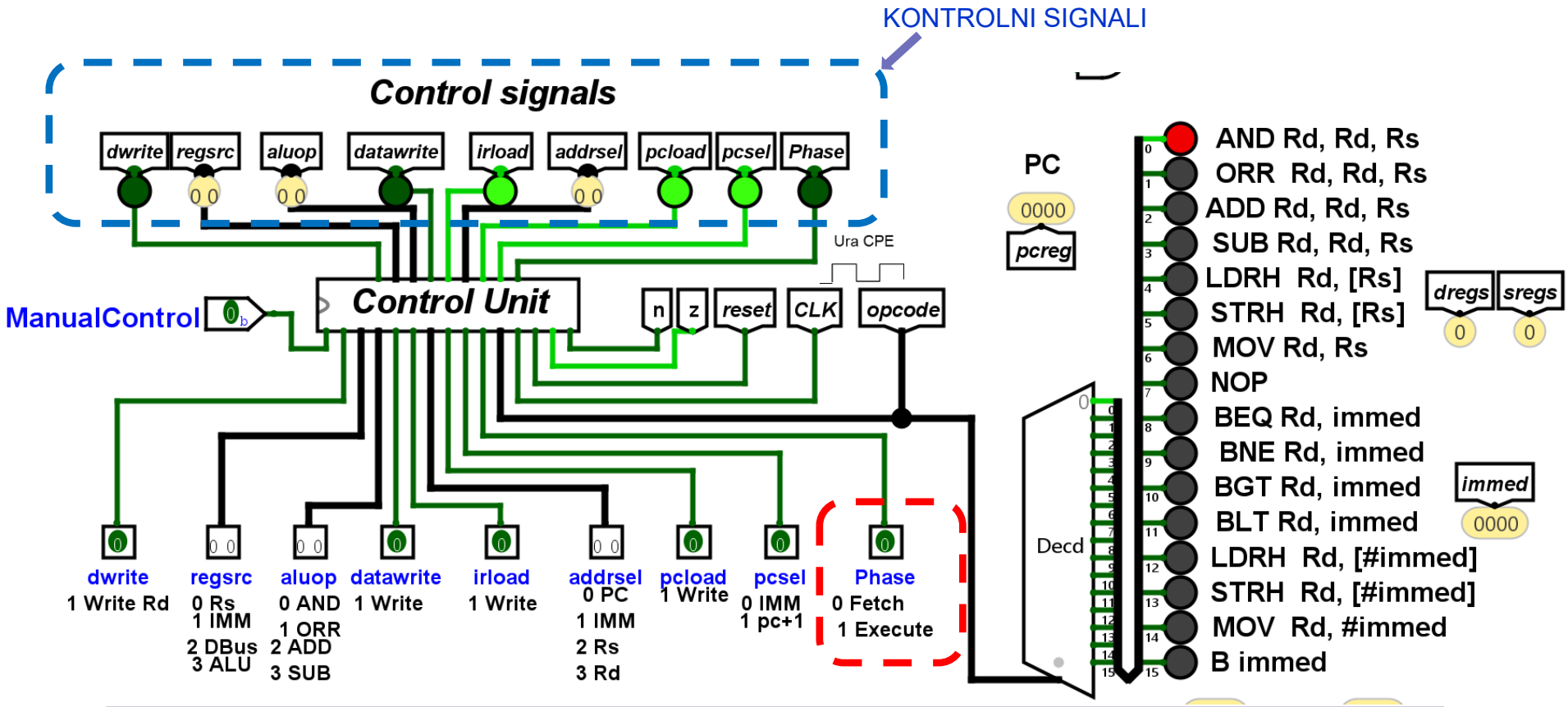
Kontrolna enota (trdo ožičena)



Kontrolna enota (trdo ožičena) : primer Mini Mimo

Strojni ukaz XXX

1. FETCH - Kontr. signali
2. EXECUTE - Kontr. signali

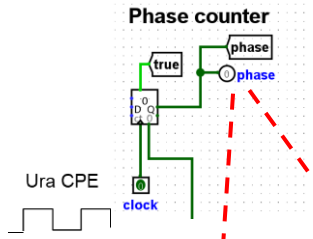


Phase = 0..FETCH, 1..EXECUTE

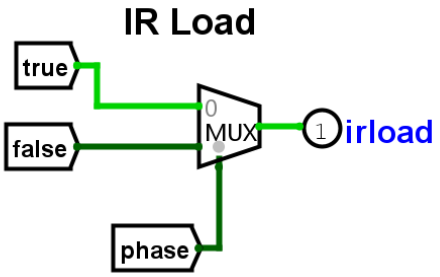
Kontrolna enota (trdo ožičena) : primer Mini Mimo

Strojni ukaz XXX

1. FETCH - Kontr. signali
2. EXECUTE - Kontr. signali

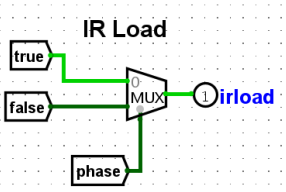


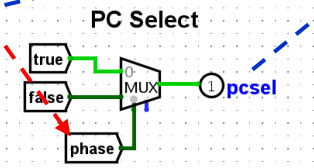
Phase = 0..FETCH,
1..EXECUTE

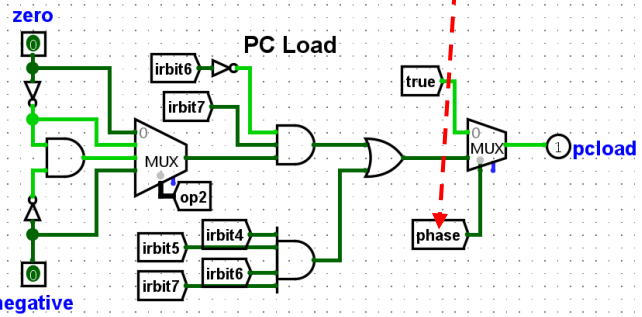


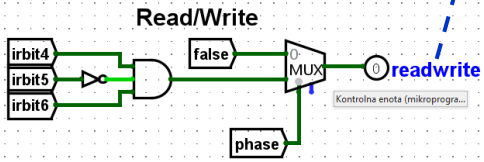
VHODI → KOMBINATORIČNA LOGIKA → IZHODI

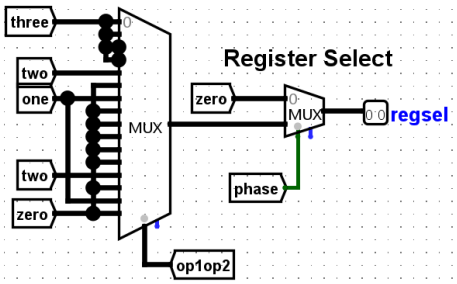
KONTROLNI SIGNALI

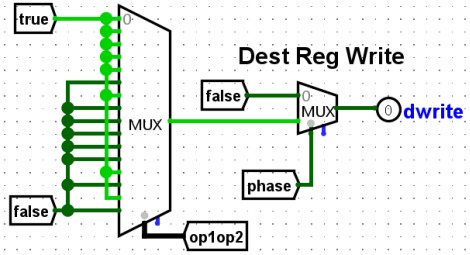


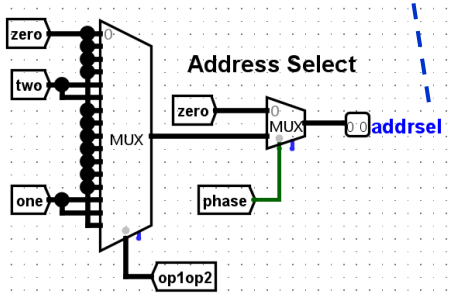






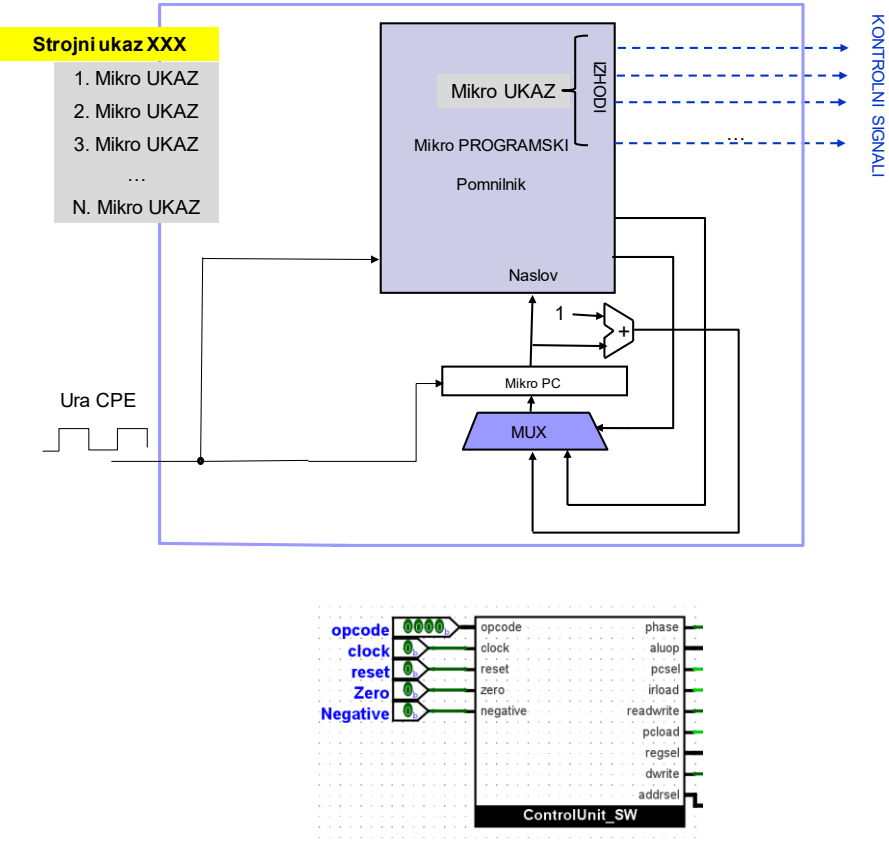




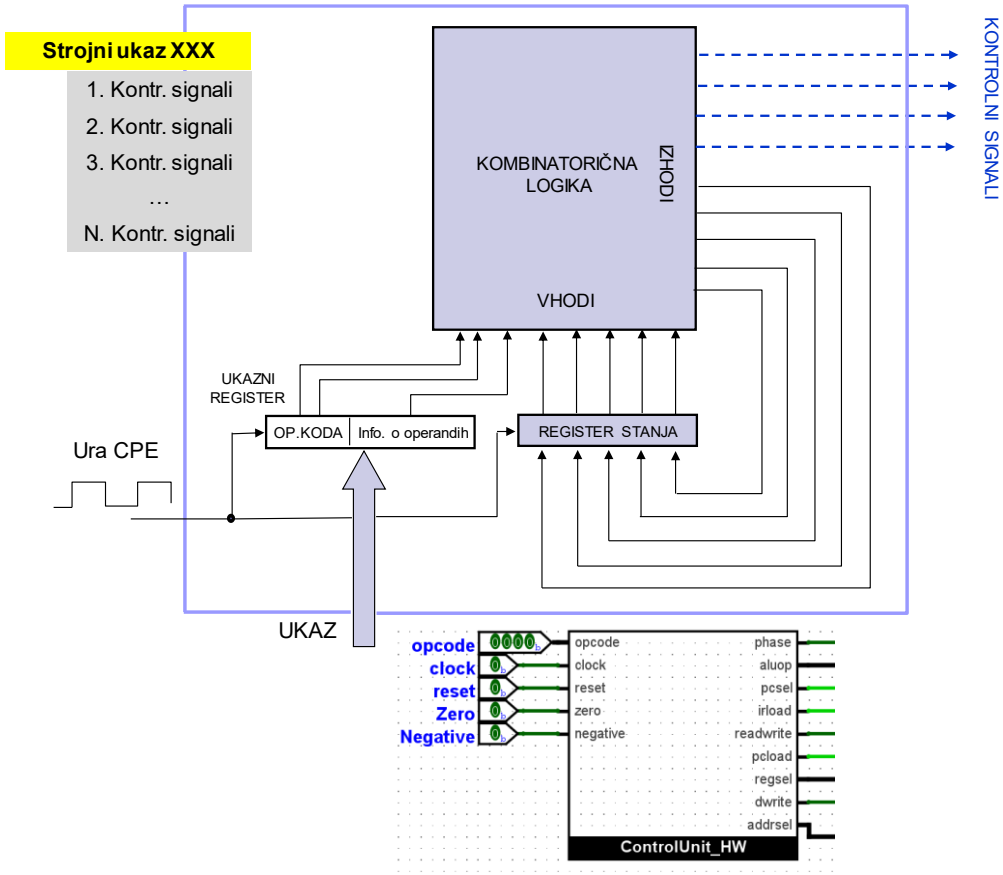


Primerjava izvedb KE

Kontrolna enota (mikroprogramska)

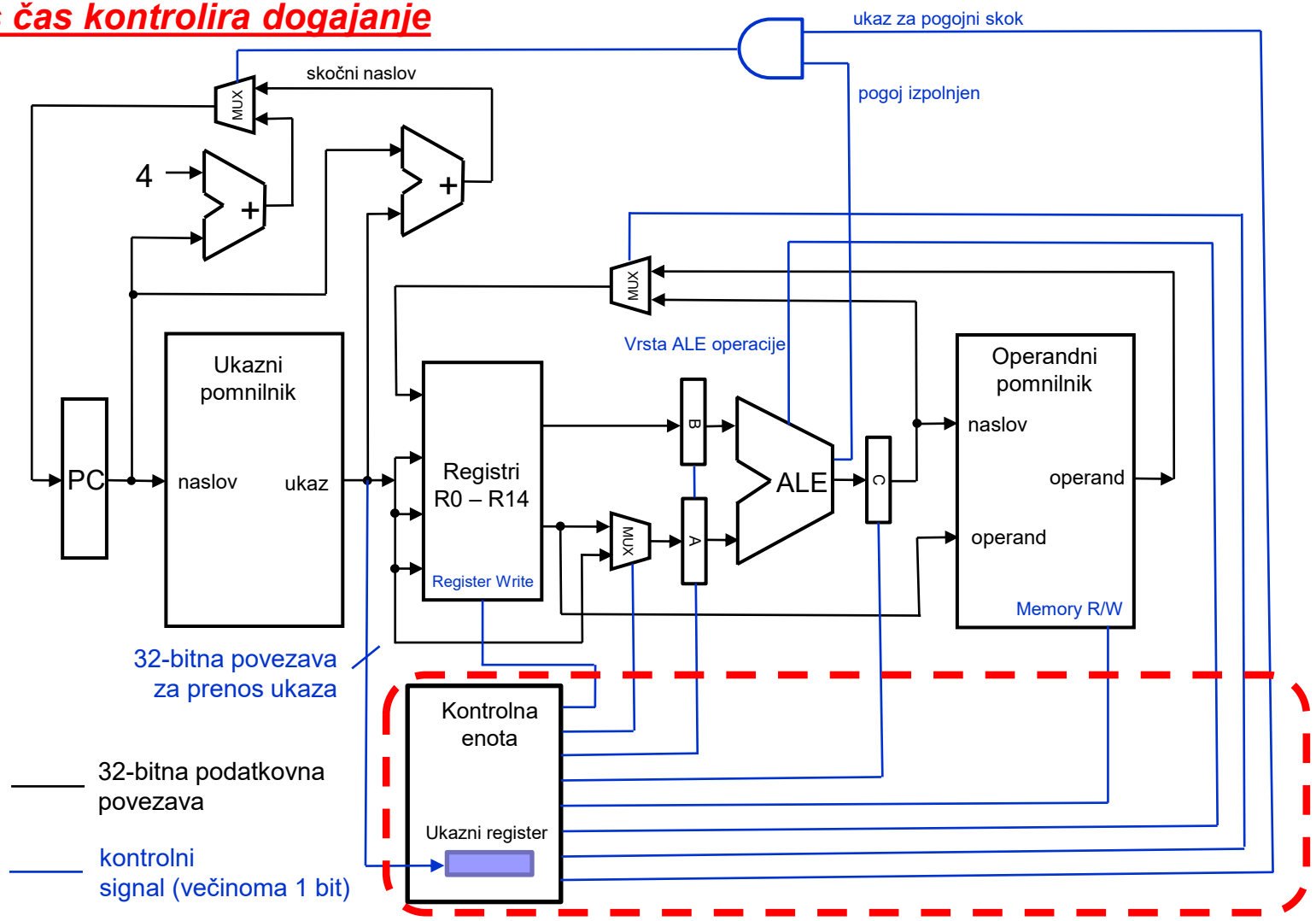


Kontrolna enota (trdo ožičena)



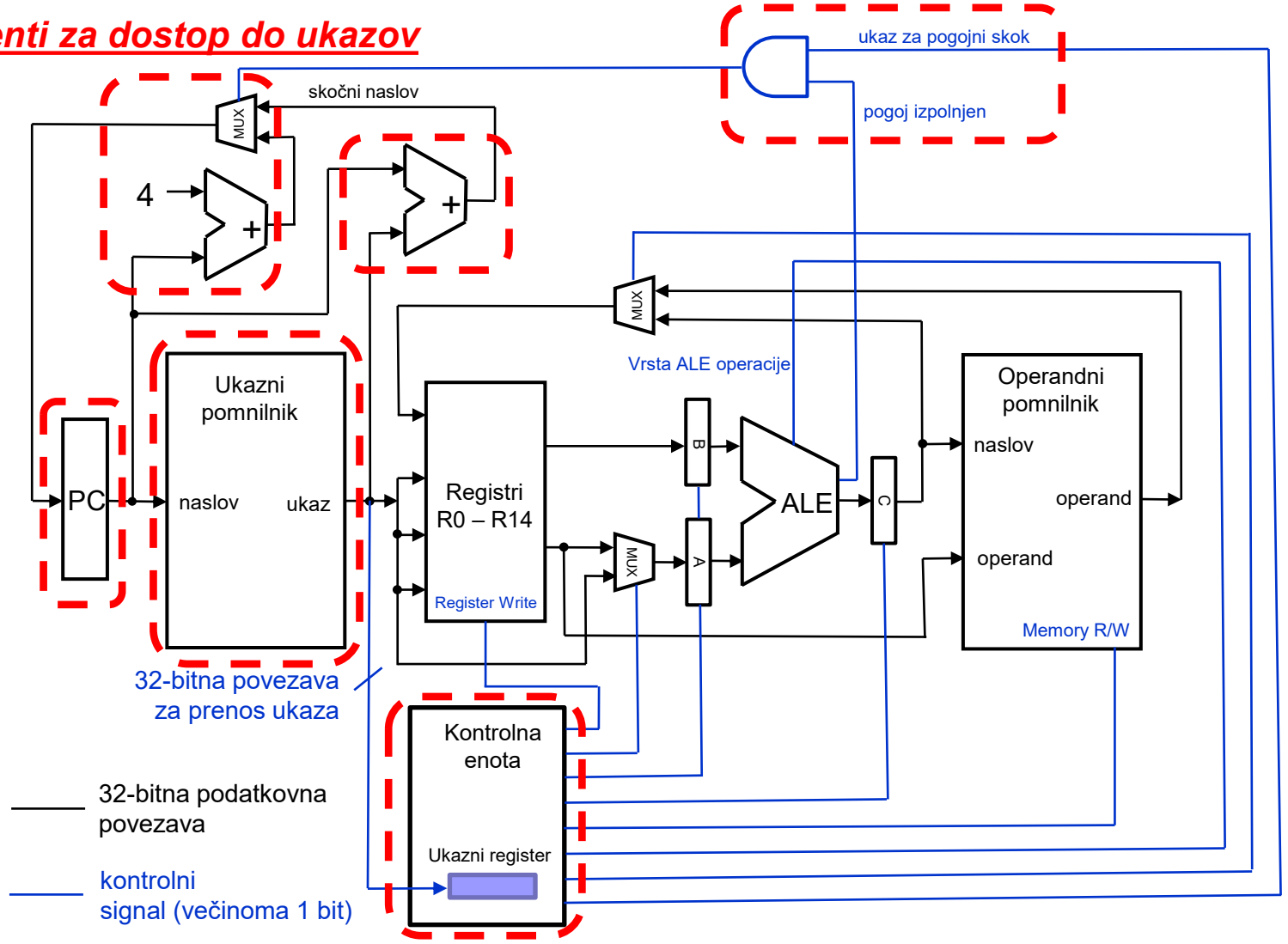
CPE s podatkovno in kontrolno enoto ter kontrolnimi signali - KE

KE ves čas kontrolira dogajanje



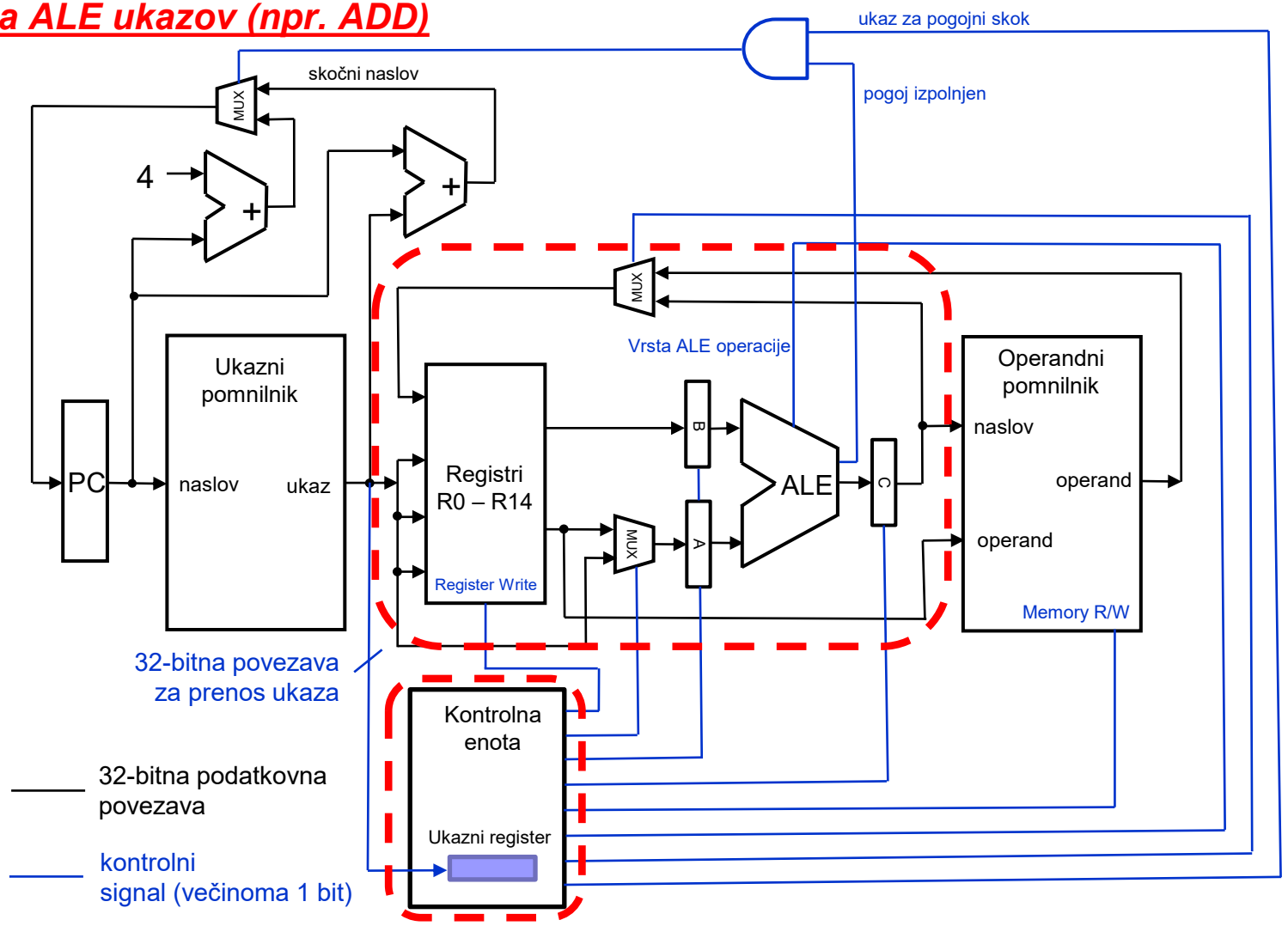
CPE s podatkovno in kontrolno enoto ter kontrolnimi signali – branje ukazov

Elementi za dostop do ukazov



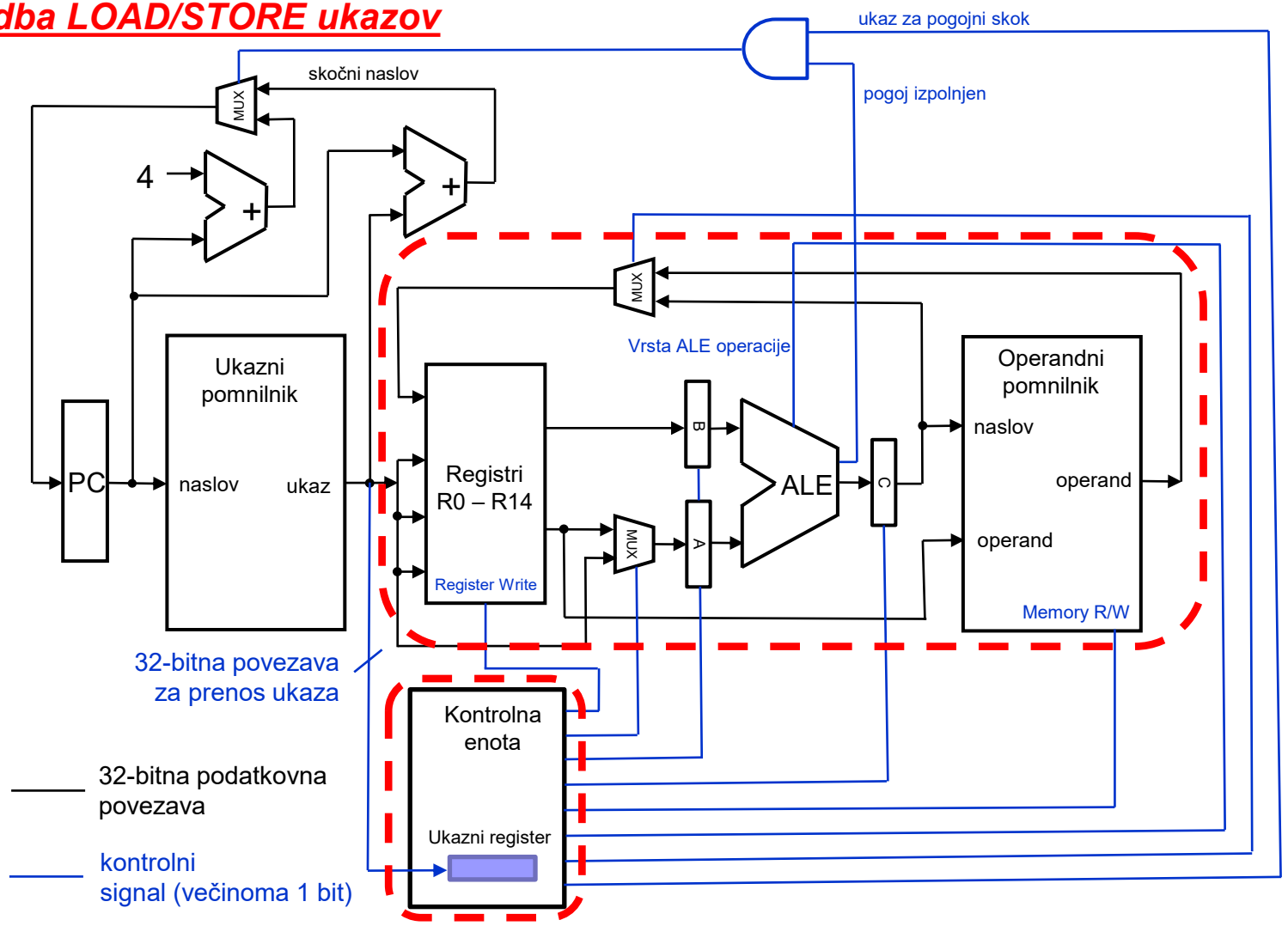
CPE s podatkovno in kontrolno enoto ter kontrolnimi signali – ALE ukazi

Izvedba ALE ukazov (npr. ADD)

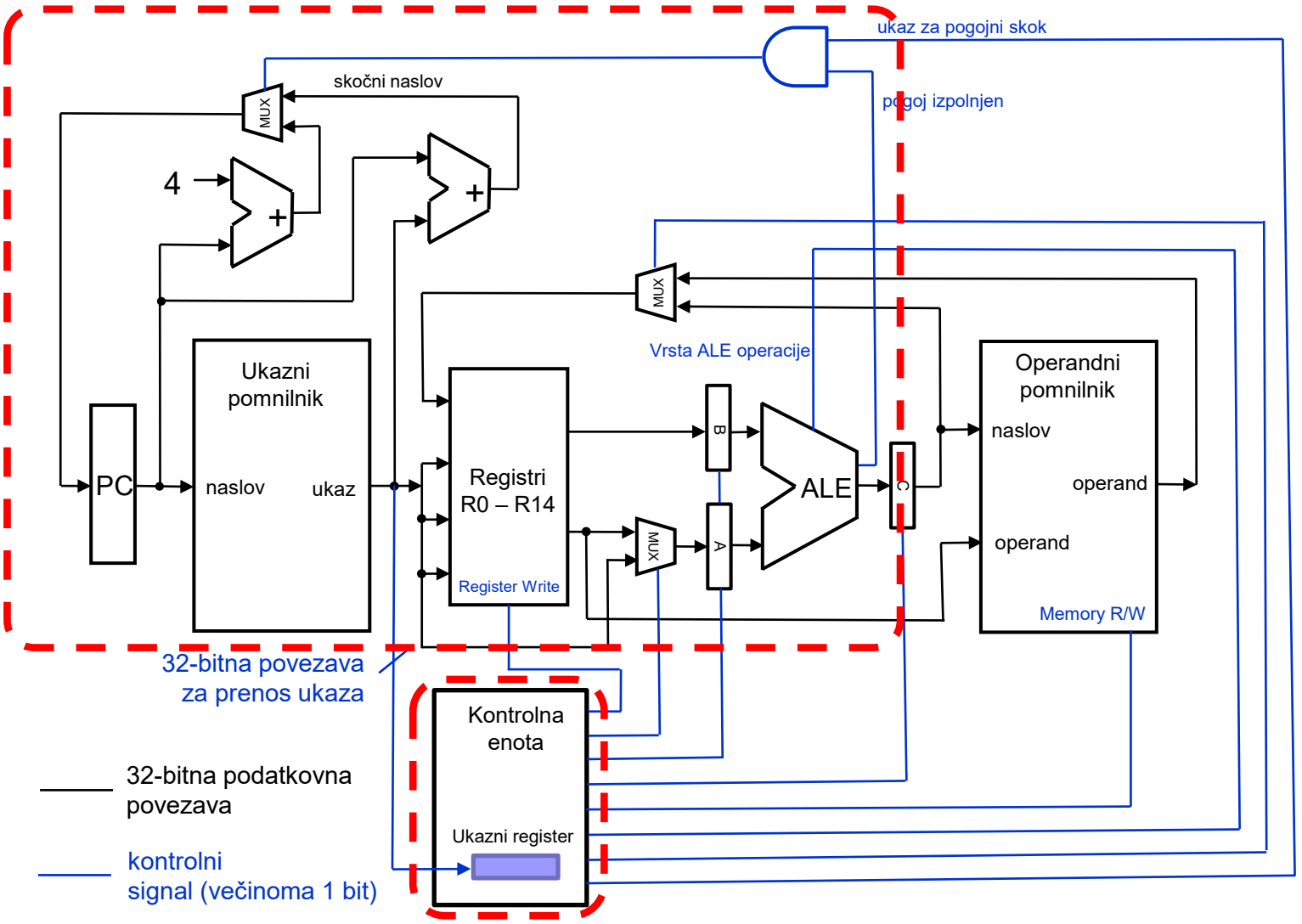


CPE s podatkovno in kontrolno enoto ter kontrolnimi signali – LOAD/STORE ukazi

Izvedba LOAD/STORE ukazov



Izvedba skočnih ukazov

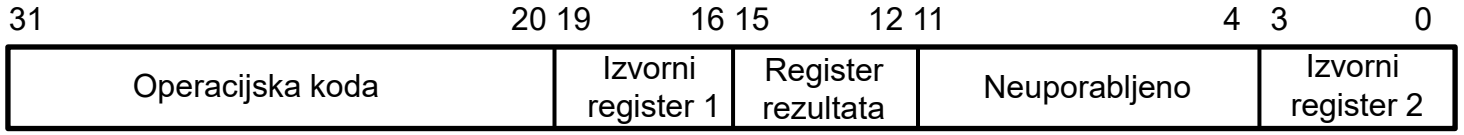


6.4 Izvajanje ukazov

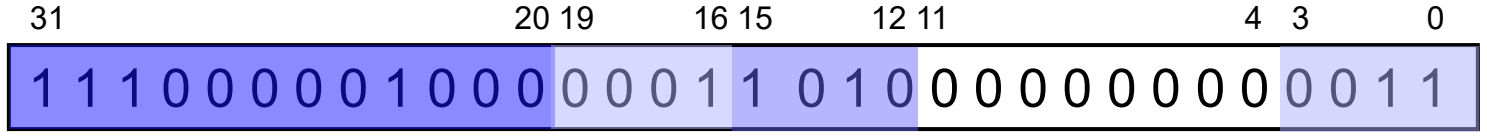
Primer izvajanja tipičnega ukaza iz skupine ALE operacij:

- ADD R10, R1, R3 R10 ← R1 + R3

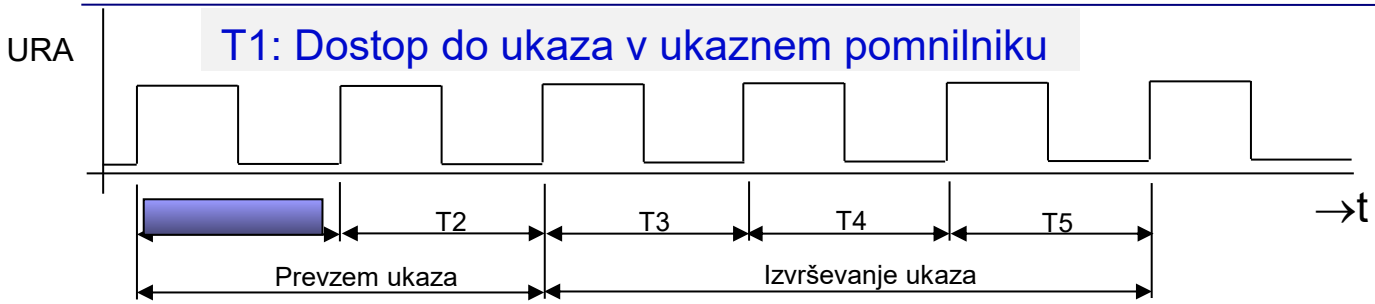
Format ukaza:



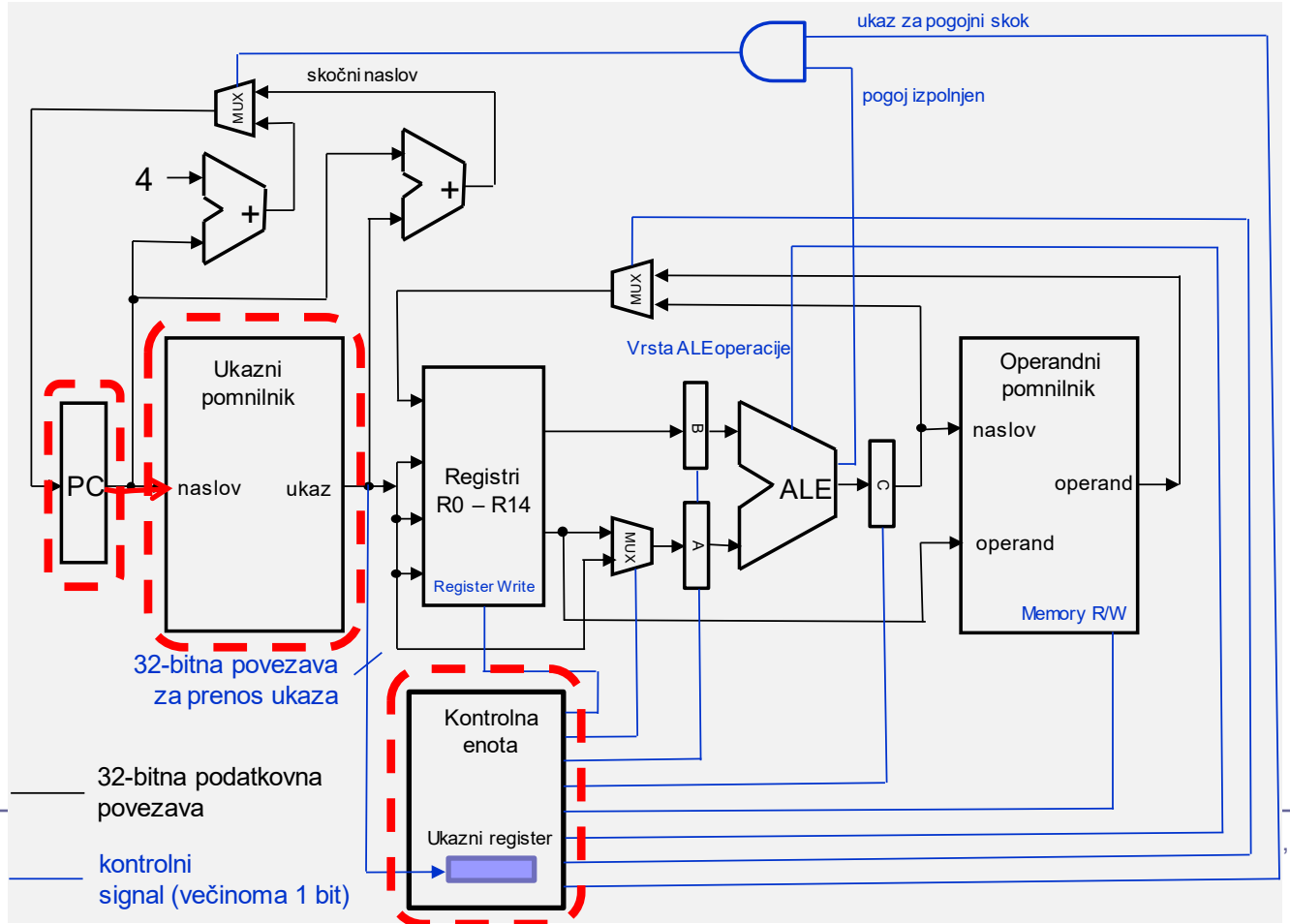
Strojni ukaz:



Izvajanje ukaza ADD: 1 elementarni korak (T1) = 1 Tcpe (urina perioda)



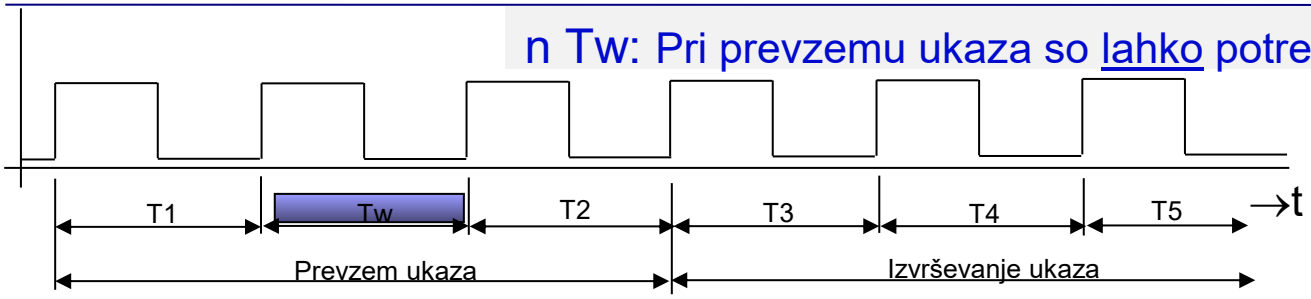
ADD R10, R1, R3



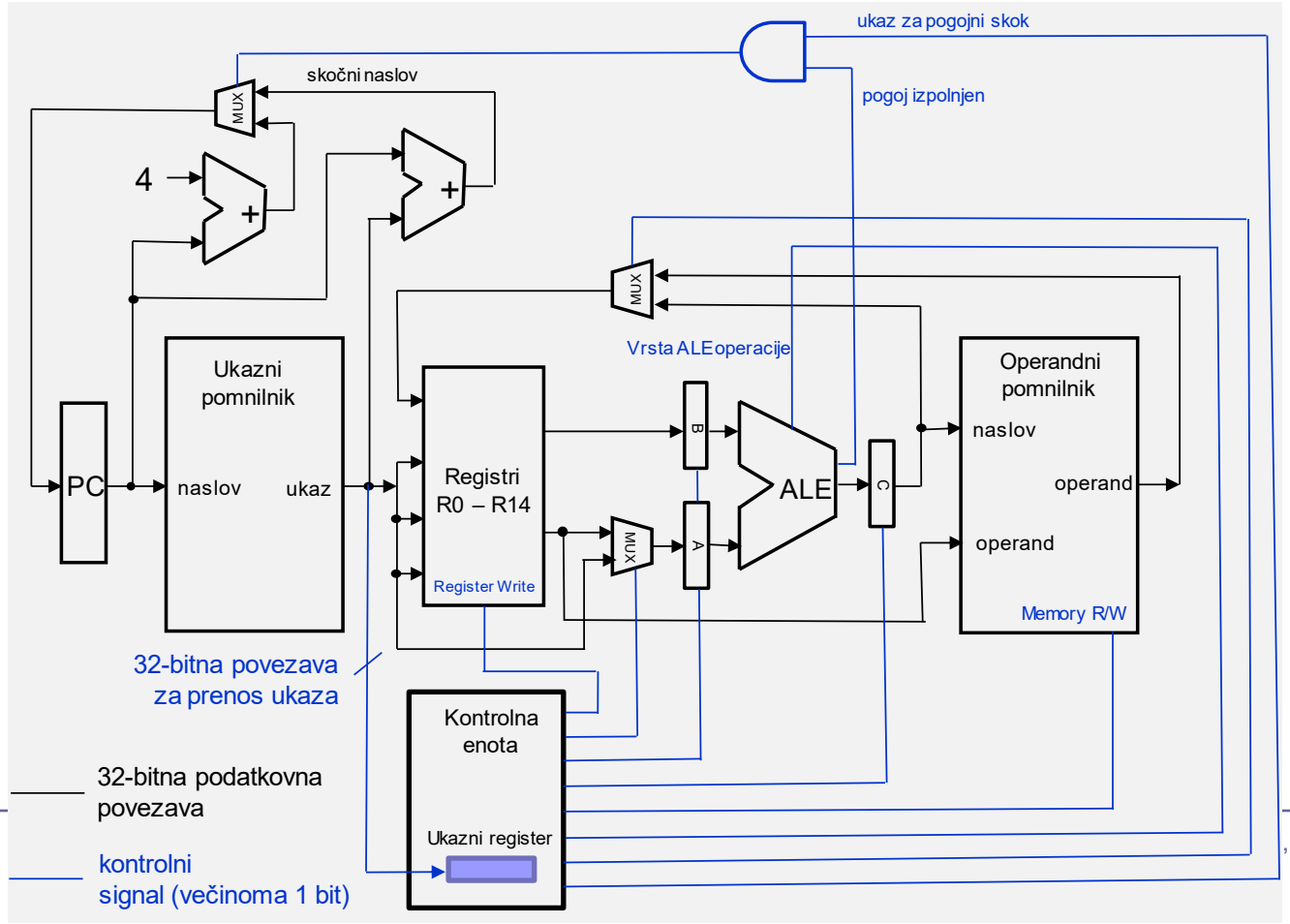
Izvajanje ukaza ADD: 2 elementarni korak (Tw) = n Tcpe (urina perioda)

URA

n Tw: Pri prevzemu ukaza so lahko potrebne čakalne urine periode

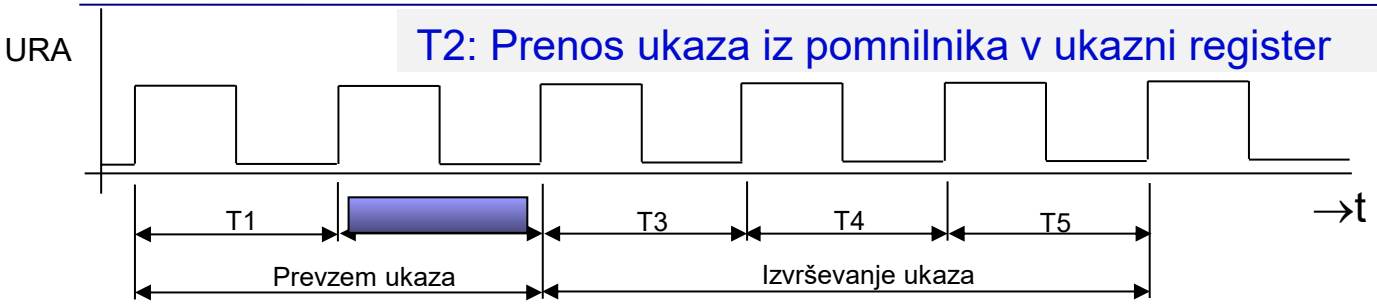


ADD R10, R1, R3

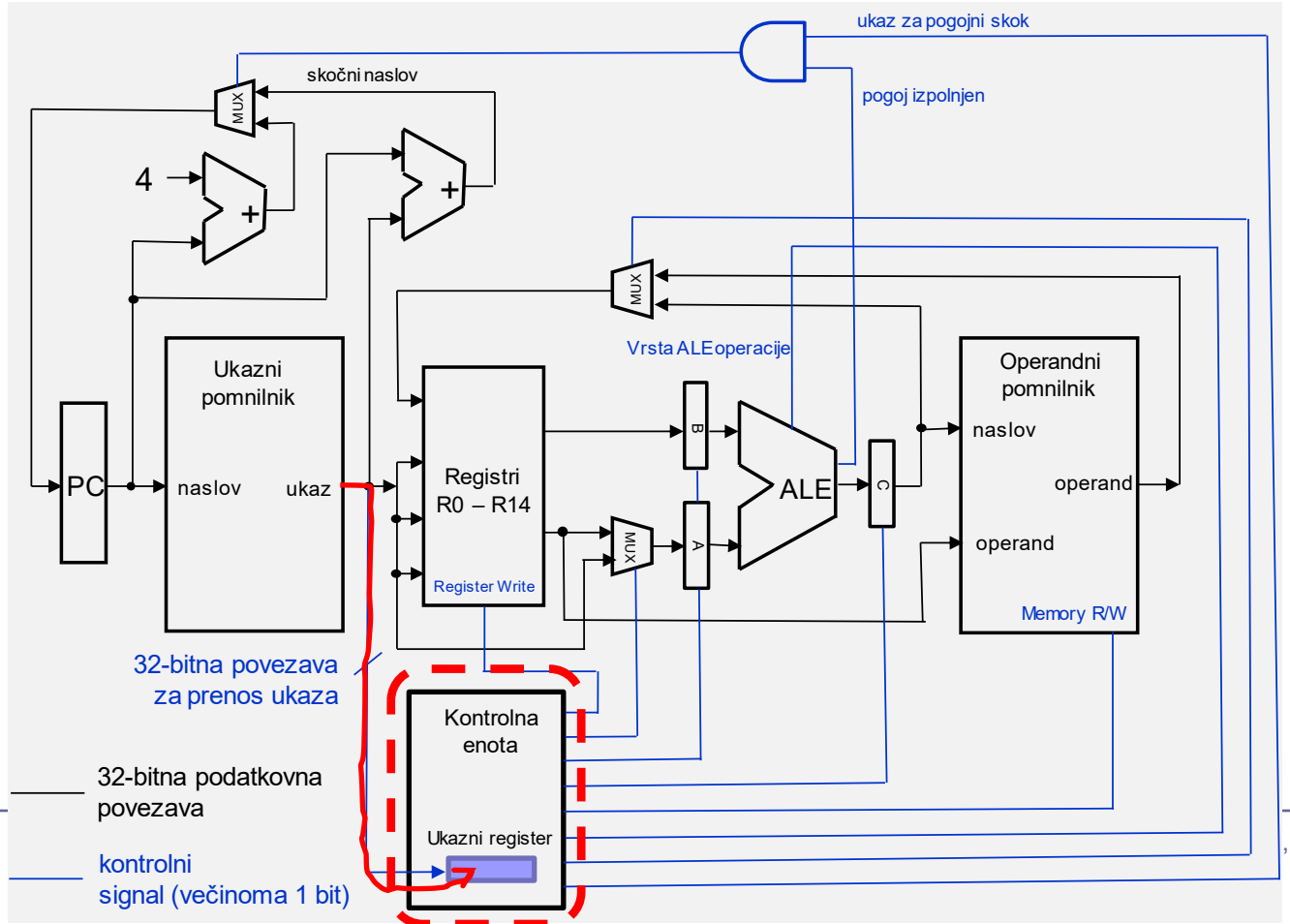


Izvajanje ukaza ADD: 2 elementarni korak (T2) = 1 Tcpe (urina perioda)

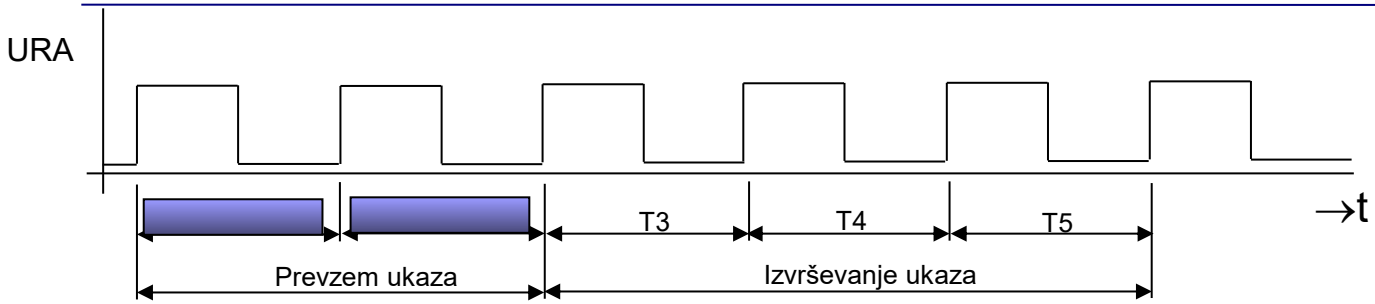
T2: Prenos ukaza iz pomnilnika v ukazni register



ADD R10, R1, R3

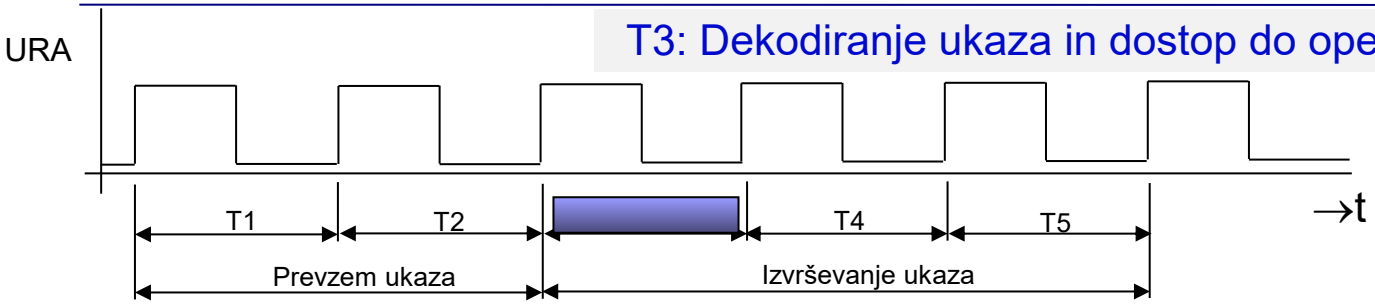


Izvajanje ukaza ADD:

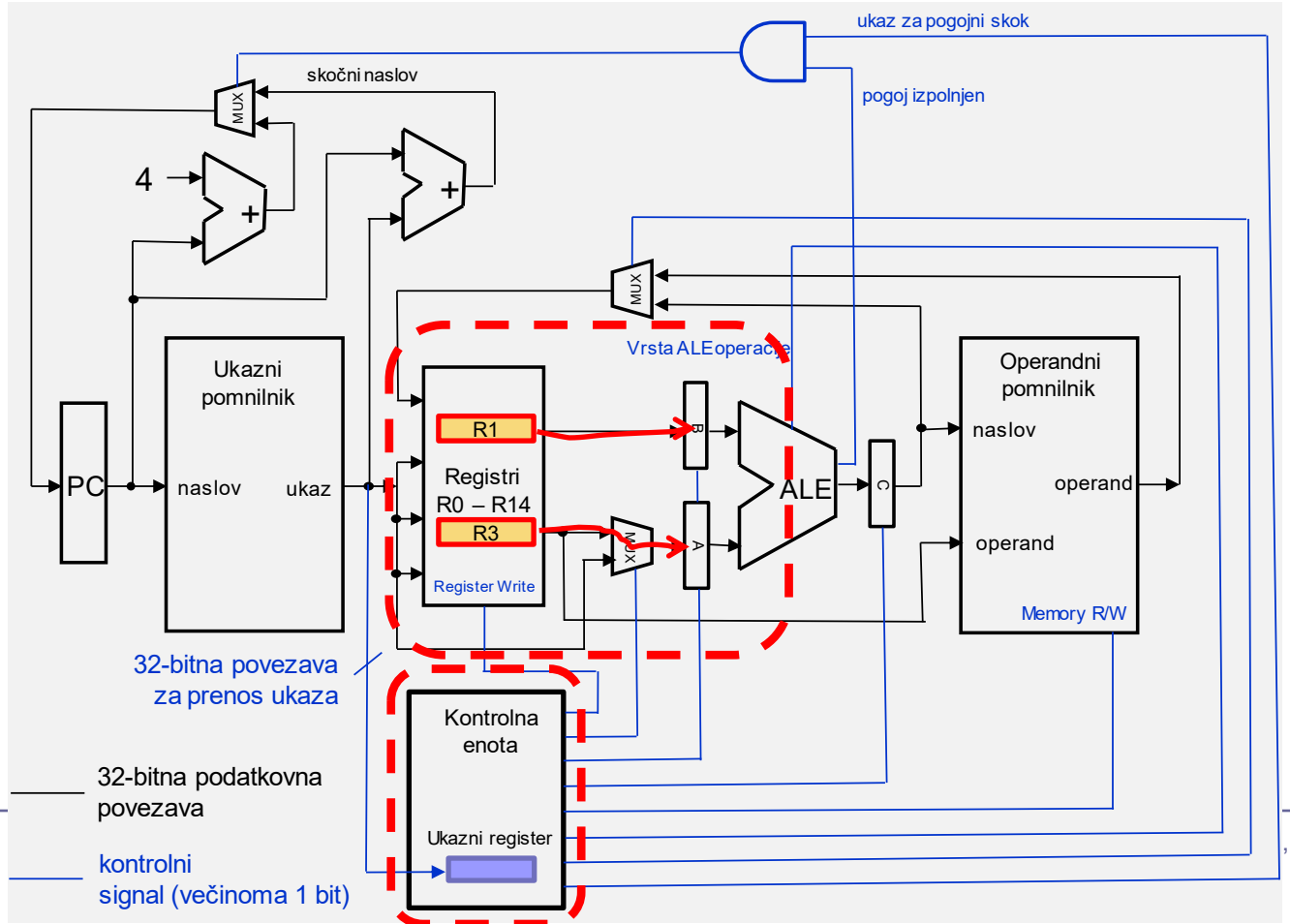


ADD R10, R1, R3

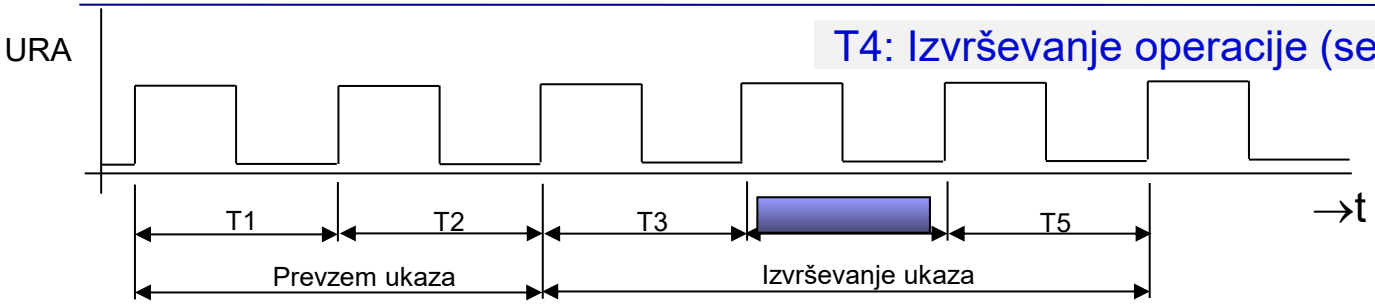
- Izvajanje ukaza ADD traja npr. 5 urinih period (CPI_{ALE}=5)
  - T1: Dostop do ukaza v pomnilniku
  - T2: Prenos ukaza iz pomnilnika v ukazni register
  - T3: Dekodiranje ukaza in dostop do operandov v registrih R1 in R3
  - T4: Izvrševanje operacije (seštevanje)
  - T5: Shranjevanje rezultata v register R10

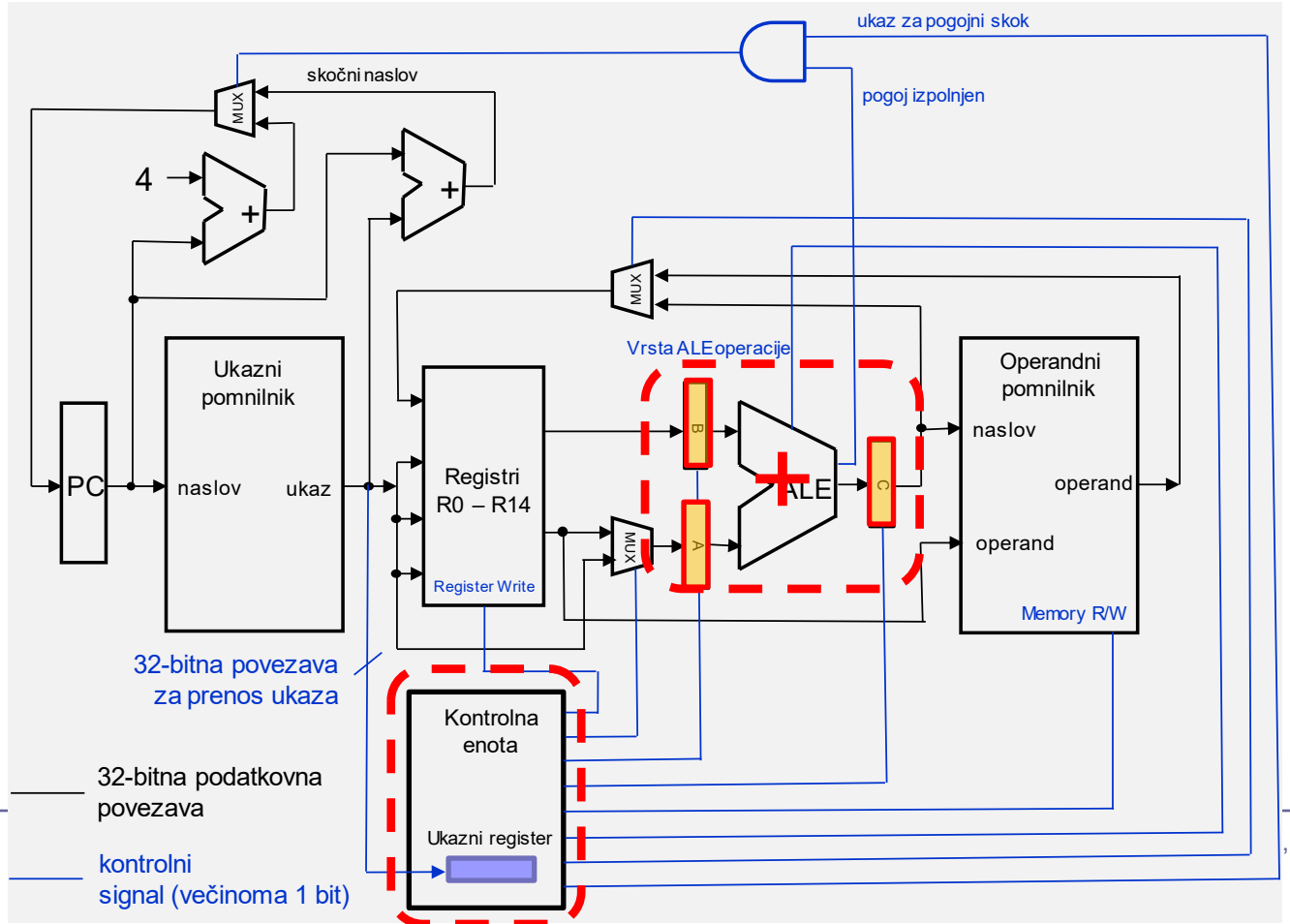

URA
T3: Dekodiranje ukaza in dostop do operandov v reg. R1 in R3
T1
T2
T4
T5
→t
Prevzem ukaza
Izvrševanje ukaza

ADD R10, R1, R3

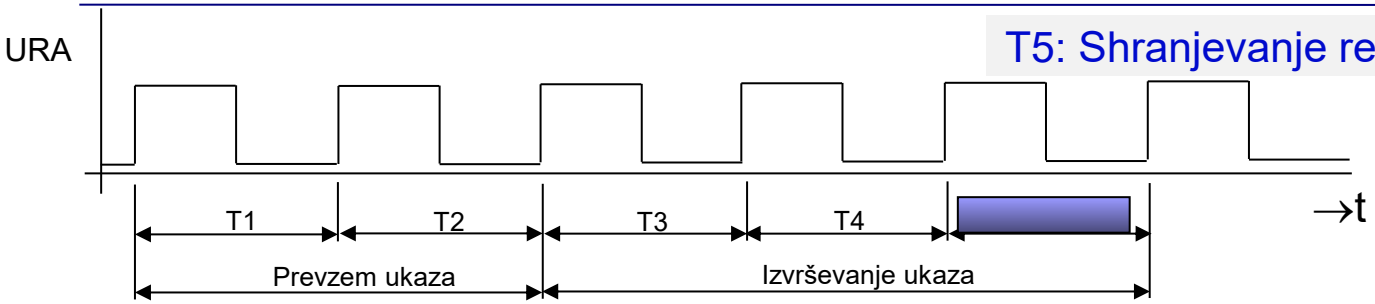

ukaz za pogojni skok
skočni naslov
pogoj izpolnjen
MUX
4
+
+
MUX
Vrsta ALE operacije
Ukazni pomnilnik
naslov
ukaz
PC
R1
Registri R0 – R14
R3
Register Write
B
A
MUX
ALE
C
Operandni pomnilnik
naslov
operand
operand
Memory R/W
32-bitna povezava za prenos ukaza
Kontrolna enota
Ukazni register
32-bitna podatkovna povezava
kontrolni signal (večinoma 1 bit)

Izvajanje ukaza ADD: 4 elementarni korak (T4) = 1 Tcpe (urina perioda)



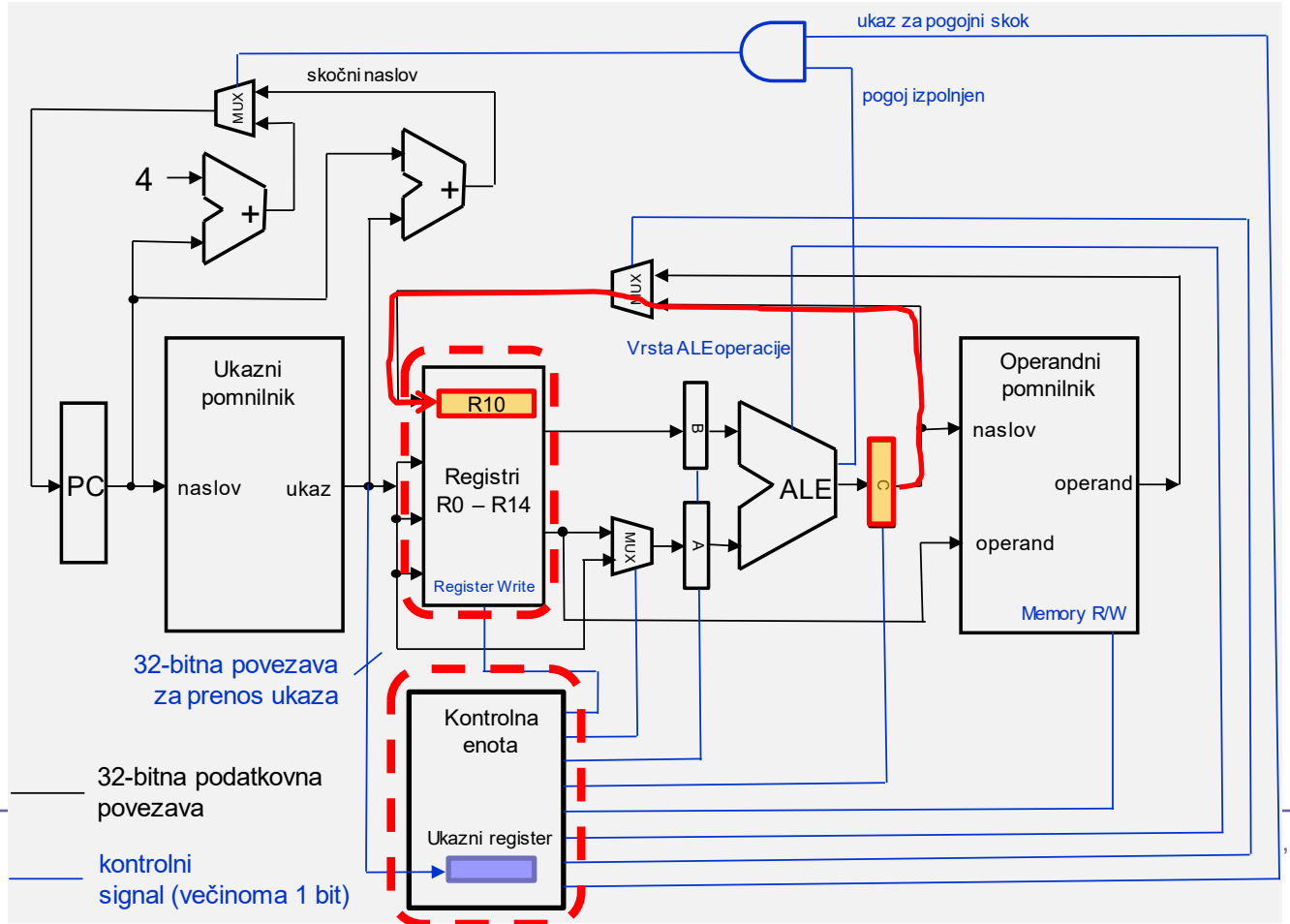


Izvajanje ukaza ADD: 5 elementarni korak (T5) = 1 Tcpe (urina perioda)

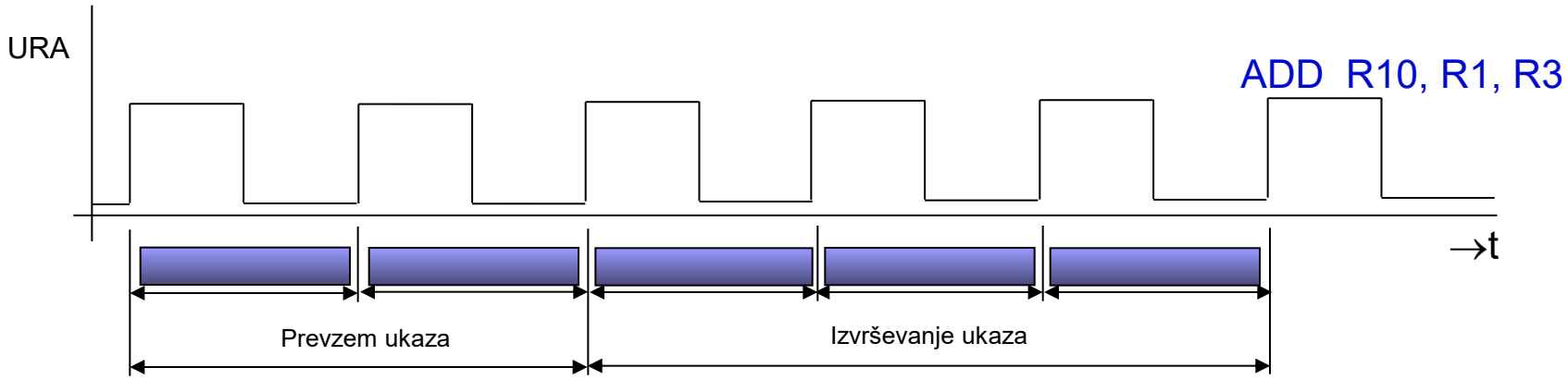


T5: Shranjevanje rezultata v register R10

ADD R10, R1, R3



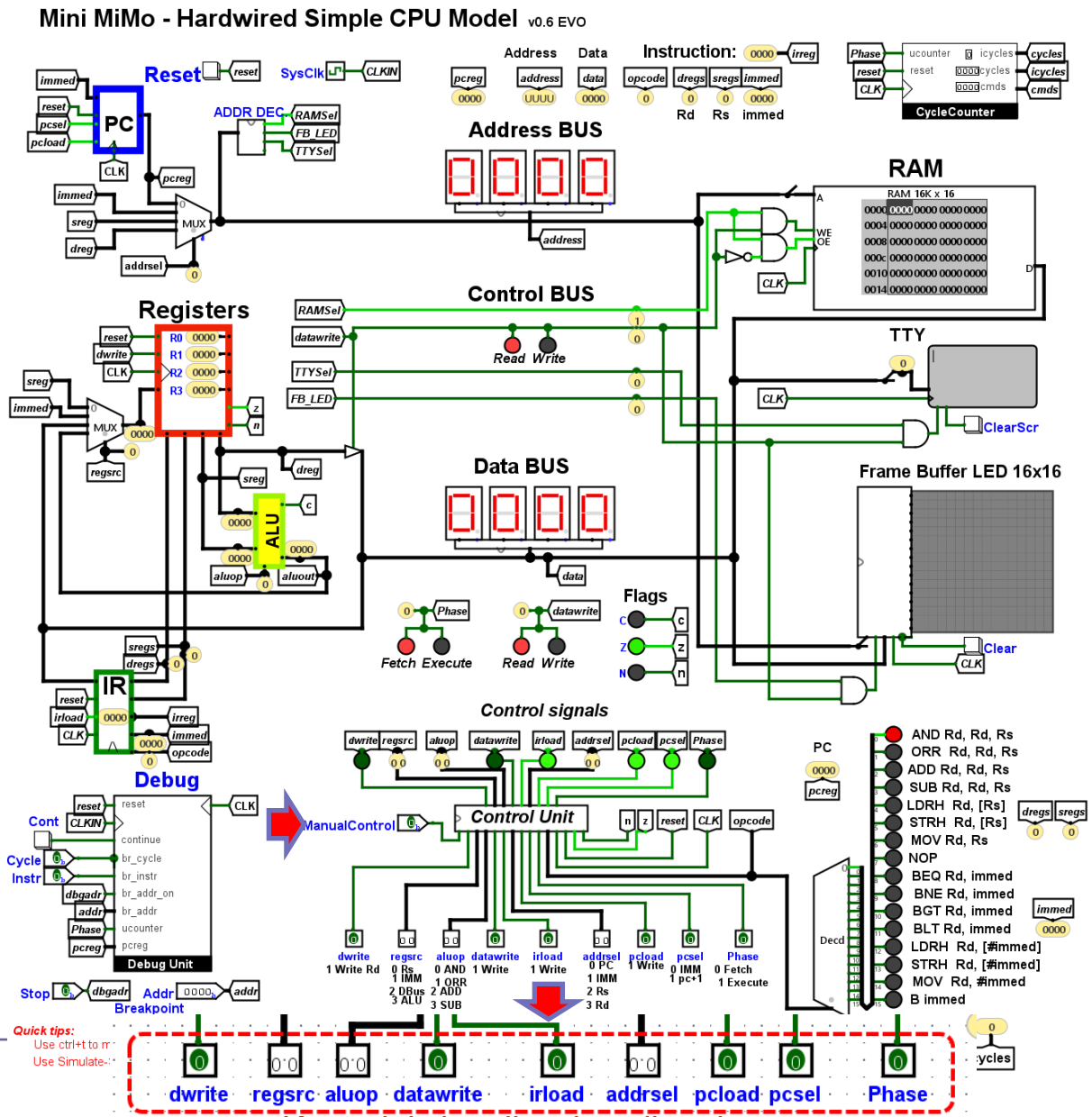
Izvajanje ukaza ADD: 1 elementarni korak (Tx) = 1 Tcpe (urina perioda)



- Izvajanje ukaza ADD traja npr. 5 urinih period (CPI_{ALE}=5)
  - T1: Dostop do ukaza v pomnilniku
  - T2: Prenos ukaza iz pomnilnika v ukazni register
  - T3: Dekodiranje ukaza in dostop do operandov v registrih R1 in R3
  - T4: Izvrševanje operacije (seštevanje)
  - T5: Shranjevanje rezultata v register R10

https://github.com/LAPSyLAB/RALab-STM32H7/tree/main/MiniMiMo_HW_CPE_Model

CPE – izvedba ukaza: primer Mini MiMo CPE



Kontrolni signali za izvedbo ukazov

https://github.com/LAPSyLAB/RALab-STM32H7/tree/main/MiniMiMo_HW_CPE_Model

Mini MiMo CPE primer – vsota dveh števil

16 bitni ukazi - format:

op1	op2	Rd	Rs	immediate
2b	2b	2b	2b	8b

Program

Naslov	Oznaka	Ukaz v zbirniku	Strojni ukaz
0x0000	main:	MOV R0, #0x20	e020
0x0001		LDRH R1, [R0]	4400
0x0002		MOV R0, #0x21	e021
0x0003		LDRH R2, [R0]	4800
0x0004		ADD R2, R2, R1	2900
0x0005		MOV R0, #0x22	e022
0x0006		STRH R2, [R0]	5800
0x0007	inf:	B inf	f007

Zbirnik v Excelu

	A	B	C	D	E
1	Address	Instruction	Rd	Rs	Immed
2	0	MOV Rd, #immec	R0	R0	32
3	1	LDRH Rd, [Rs]	R1	R0	
4	2	MOV Rd, #immec	R0	R0	33
5	3	LDRH Rd, [Rs]	R2	R0	
6	4	ADD Rd, Rd, Rs	R2	R1	
7	5	MOV Rd, #immec	R0	R0	34
8	6	STRH Rd, [Rs]	R2	R0	
9	7	B immed	R0	R0	7
10	8				
11	9				

I
Strojni ukaz
E020
4400
E021
4800
2900
E022
5800
F007

Register

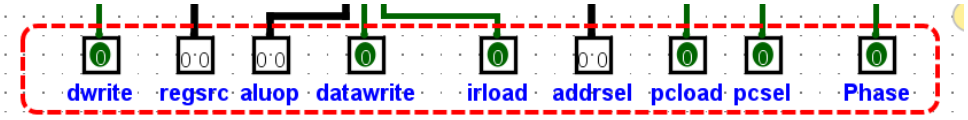
minimimo_vsota.ram | test17-tournament.s | minimimo_sestej.ram

```
1  v3.0 hex words addressed
2  0000: e020 4400 e021 4800 2900 e022 5800 f007
3  0010: 0000 0000 0000 0000 0000 0000 0000 0000
4  0020: 0010 0040 0000 0000 0000 0000 0000 0000
```

Kontrolna enota

Kontrolni signali za izvedbo ukazov

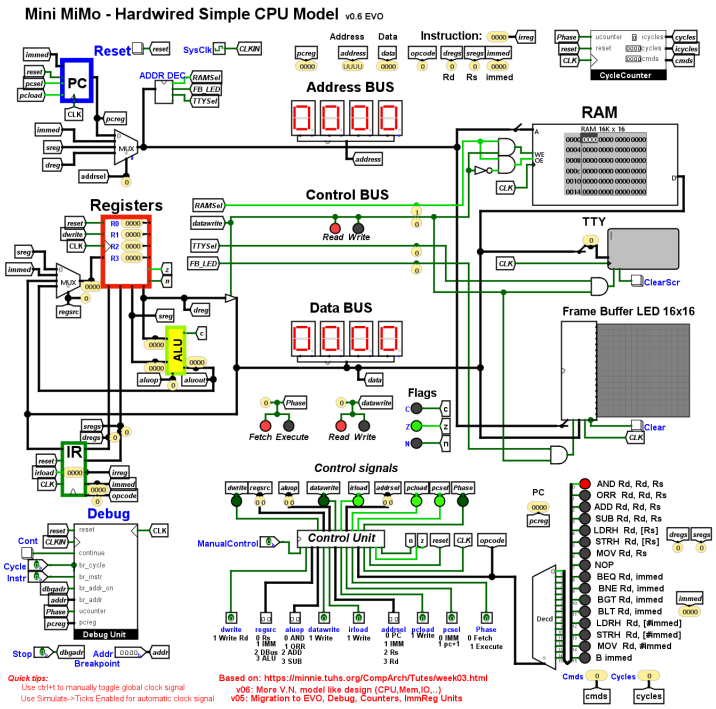
op1	op2	ARM9 zapis	pc sel	pc load	ir load	rw	dwrite	addr sel	reg sel	d reg	s reg	aluop
xx	xx	FETCH - vsi ukazi	1(pc+1)	1	1	0	0	0(pc)	x	x	x	x
00	10	ADD Rd, Rd, Rs	x	0	0	0	1	x	3(ALU)	Rd	Rs	op2=0b10
01	00	LDRH Rd, [Rs]	x	0	0	0	1	2(Rs)	2(Dbus)	Rd	Rs	x
01	01	STRH Rd, [Rs]	x	0	0	1	0	2(Rs)	x	Rd	Rs	x
11	10	MOV Rd, #immed	x	0	0	0	1	x	1 (IM)	Rd	x	x
11	11	B immed	0(IM)	1	0	0	0	x	x	x	x	x



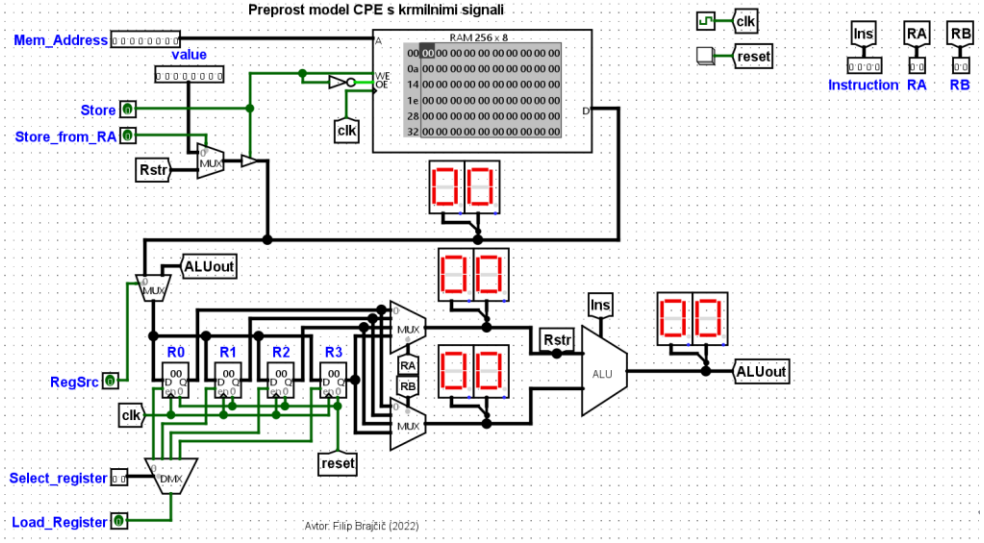
Mini MiMo CPE RAM pomnilnik

Izziv (DN1 ali neobvezni dodatek k DN1)

Delo na Mini MiMo modelu



Izziv: Si upate narediti svoj procesor ?



Primer lastne CPE 22/23 (avtor: F.Brajčič)

https://github.com/LAPSyLAB/RALab-STM32H7/tree/main/MiniMiMo_HW_CPE_Model
https://github.com/LAPSyLAB/RALab-STM32H7/tree/main/LogisimEVO_vezja/Prispevki

6.5 Paralelno izvajanje ukazov

- Običajna zgradba CPE – izvajanje strojnega ukaza traja najmanj 3 ali 4 urine periode, običajno pa tudi več.
- Povprečno število ukazov, ki jih CPE izvede v eni sekundi (*IPS – Instructions Per Second*):

$$IPS = \frac{f_{CPE}}{CPI}$$

IPS je zelo veliko število, zato ga delimo z 10^6 in dobimo MIPS

$$MIPS = \frac{f_{CPE}}{CPI \cdot 10^6}$$

MIPS = Million Instructions per Second

f_{CPE} = frekvenca CPE ure

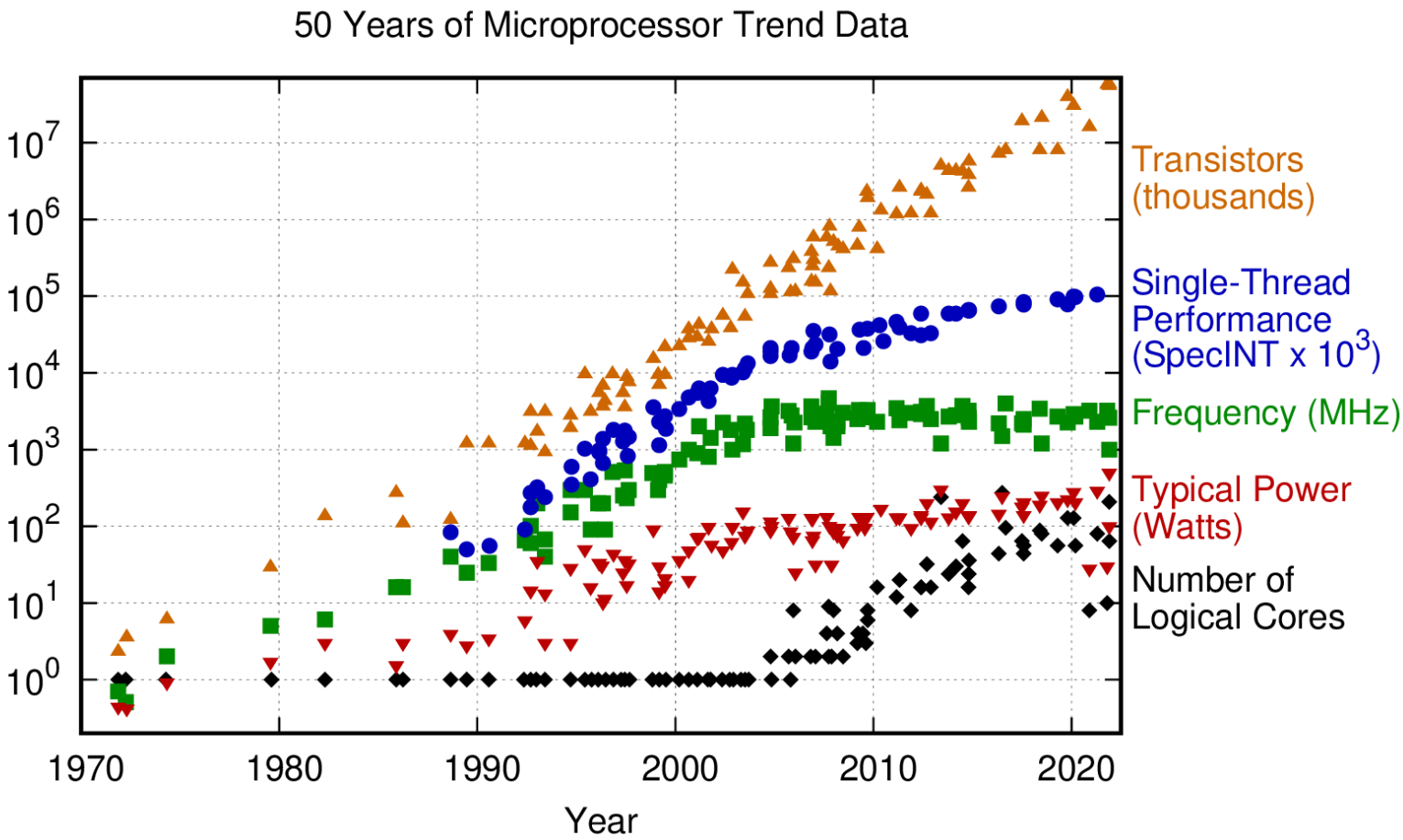
CPI = Cycles per Instruction (povprečno število urinih period za izvedbo enega ukaza)

MIPS

- MIPS - število ukazov, ki jih CPE izvede v eni sekundi, lahko povečamo tako, da povečamo ali f_{CPE} in/ali zmanjšamo CPI:

$$\uparrow MIPS = \frac{\uparrow f_{CPE}}{\downarrow CPI \cdot 10^6}$$

  - Z uporabo hitrejših elementov (povečanje f_{CPE} = več urinih period v sekundi)
  - Z uporabo večjega števila elementov dosežemo zmanjšanje *CPI (*manj urinih period za izvedbo enega ukaza), ker se več operacij izvede v eni urini periodi
  - Uporaba hitrejših elementov ne omogoča velikega povečanja hitrosti, s seboj pa prinaša še druge probleme.



Vir: https://raw.githubusercontent.com/karlrupp/microprocessor-trend-data/master/50yrs/50-years-processor-trend.png

Povečevanje števila tranzistorjev – Moorov zakon

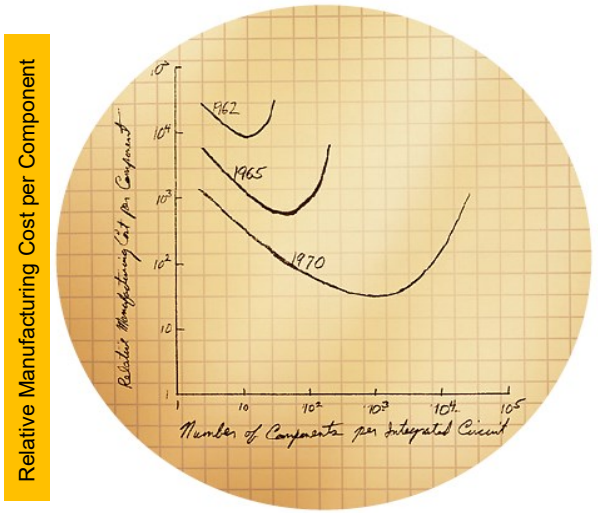
- Revija Electronic Magazine je leta 1965 objavila članek Gordona E. Moora, v katerem je napovedal, da se bo število tranzistorjev, ki so jih proizvajalci sposobni izdelati na čipu, podvojilo vsako leto.

- Leta 1975 je napoved popravil, da se bo število tranzistorjev podvojilo na vsaki dve leti.

- Kar je bilo takrat mišljeno kot izkustveno pravilo in naj bi veljalo naslednjih nekaj let, velja še danes in je poznano kot Moorov zakon.

intel

Moore's Law



Number of Components per IC

In 1965, Gordo he pace of silicon technology. Decades later, Moore's Law remains true, driven largely by Intel's unparalleled silicon expertise.

According to Moore's Law, the number of transistors on a chip roughly doubles every two years. As a result the scale gets smaller and smaller. For decades, Intel has met this formidable challenge through investments in technology and manufacturing resulting in the unparalleled silicon expertise that has made Moore's Law a reality.

- Gordon E. Moore (2023) je bil častni predsednik Intela, v letu 1968 pa je bil soustanovitelj in izvršni podpredsednik Intela.

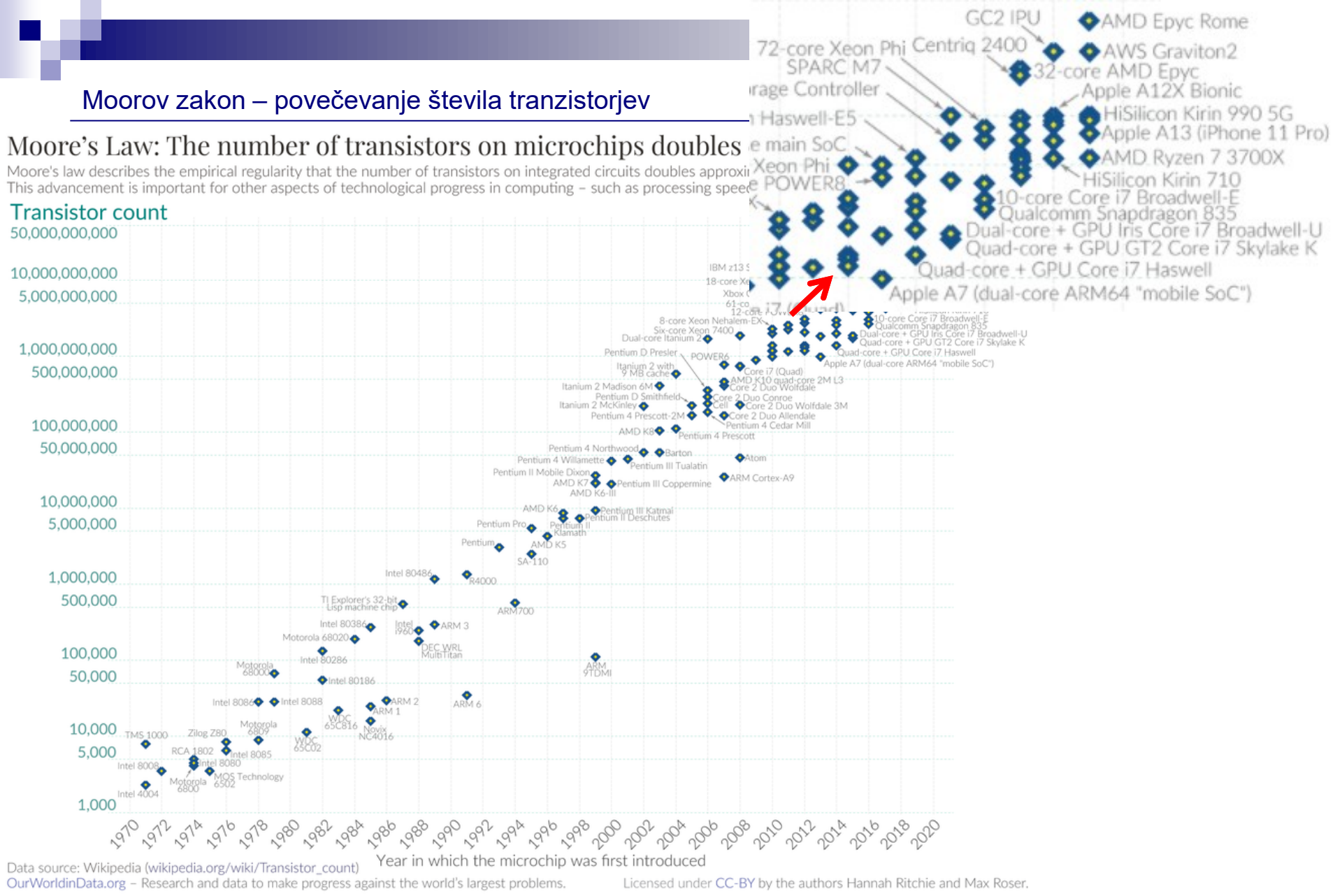
- Pri isti tehnologiji se je v obdobju 20 let pred nekaj časa najvišja hitrost logičnih elementov povečala za približno 10-krat.

- V istem času se je največje število elementov na enem čipu povečalo za približno 500 do celo 5000-krat pri pomnilniških čipih.

Povečevanje števila tranzistorjev – Moorov zakon

- Electronic Magazine (1965):
  - Gordon E. Moore (soustanovitelj in izvršni predsednik Intela): število tranzistorjev, ki so jih proizvajalci sposobni izdelati na čipu, se bo podvojilo vsako leto.
- Leta 1975 popravek;
  - *Število tranzistorjev se bo podvojilo na vsaki dve leti.*
- Napoved „velja“ še danes in je poznana kot Moorov zakon.

Moorov zakon – povečevanje števila tranzistorjev



Moorov zakon – število tranzistorjev skozi čas - mikroprocesorji

Processor	Transistor count	Year	Designer	Process (nm)	Area (mm^2)	Transistor density ($tr./mm^2$)
MP944 (20-bit, 6-chip, 28 chips total)	74,442 (5,360 excl. ROM & RAM)[14][15]	1970[12][a]	Garrett AiResearch	?	?	?
Intel 4004 (4-bit, 16-pin)	2,250	1971	Intel	10,000 nm	12 mm^2	188
TMX 1795 (8-bit, 24-pin)	3,078[16]	1971	Texas Instruments	?	30.64 mm^2	100.5
Intel 8008 (8-bit, 18-pin)	3,500	1972	Intel	10,000 nm	14 mm^2	250
NEC μCOM-4 (4-bit, 42-pin)	2,500[17][18]	1973	NEC	7,500 nm[19]	?	?
Toshiba TLCS-12 (12-bit)	11,000+[20]	1973	Toshiba	6,000 nm	32.45 mm^2	340+
Intel 4040 (4-bit, 16-pin)	3,000	1974	Intel	10,000 nm	12 mm^2	250
Motorola 6800 (8-bit, 40-pin)	4,100	1974	Motorola	6,000 nm	16 mm^2	256
Intel 8080 (8-bit, 40-pin)	6,000	1974	Intel	6,000 nm	20 mm^2	300
TMS 1000 (4-bit, 28-pin)	8,000[b]	1974[21]	Texas Instruments	8,000 nm	11 mm^2	730
MOS Technology 6502 (8-bit, 40-pin)	4,528[c][22]	1975	MOS Technology	8,000 nm	21 mm^2	216
Intersil IM6100 (12-bit, 40-pin; clone of PDP-8)	4,000	1975	Intersil	?	?	?
CDP 1801 (8-bit, 2-chip, 40-pin)	5,000	1975	RCA	?	?	?
RCA 1802 (8-bit, 40-pin)	5,000	1976	RCA	5,000 nm	27 mm^2	185
Zilog Z80 (8-bit, 4-bit ALU, 40-pin)	8,500[d]	1976	Zilog	4,000 nm	18 mm^2	470
Intel 8085 (8-bit, 40-pin)	6,500	1976	Intel	3,000 nm	20 mm^2	325

...

Apple A17	19,000,000,000 [187]	2023	Apple	3 nm	103.8 mm^2	183,044,315
Sapphire Rapids quad-chip module (up to 60 cores and 112.5 MB of cache)[188]	44,000,000,000–48,000,000,000[189]	2023	Intel	10 nm ESF (Intel 7)	1,600 mm^2	27,500,000–30,000,000
Apple M2 Pro (12-core 64-bit ARM64 SoC, SIMD, caches)	40,000,000,000[190]	2023	Apple	5 nm	?	?
Apple M2 Max (12-core 64-bit ARM64 SoC, SIMD, caches)	67,000,000,000[190]	2023	Apple	5 nm	?	?
Apple M2 Ultra (two M2 Max dies)	134,000,000,000[6]	2023	Apple	5 nm	?	?
AMD Epyc Bergamo (4th gen/97X4 series) 9-chip module (up to 128 cores and 256 MB (L3) + 128 MB (L2) cache)	82,000,000,000[191]	2023	AMD	5 nm (CCD) 6 nm (IOD)	?	?
AMD Instinct MI300A (multi-chip module, 24 cores, 128 GB GPU memory + 256 MB (LLC/L3) cache)	146,000,000,000[192][193]	2023	AMD	5 nm (CCD, GCD) 6 nm (IOD)	1,017 mm^2	144,000,000
Processor	**Transistor count**	**Year**	**Designer**	**Process (nm)**	**Area (mm^2)**	**Transistor density ($tr./mm^2$)**

Trenutno število tranzistorjev v različnih napravah

Year	Component	Name	Number of MOSFETs (in trillions)	Remarks
2022	Flash memory	Micron's V-NAND module	5.3	stacked package of sixteen 232-layer 3D NAND dies
2020	any processor	Wafer Scale Engine 2	2.6	wafer-scale design of 84 exposed fields (dies)
2024	GPU	Nvidia B100	0.208 [Brez naslova]	Uses two reticle limit dies, with 104 billion transistors each, joined together and acting as a single large monolithic piece of silicon
2023	microprocessor (commercial)	M2 Ultra	0.134	SoC using two dies joined together with a high-speed bridge
2020	DLP	Colossus Mk2 GC200	0.059	An IPU[clarification needed] in contrast to CPU and GPU

Slo oznaka enote:
bilijon (tisoč milijard)

DLP… „Deep learning processor“

Kako učinkovito izkoristimo več elementov ?

- Učinkovito povečanje hitrosti CPE:
  - CPE paralelno izvaja več funkcij, to pa pomeni povečanje števila logičnih elementov.

Paralelizem lahko izkoristimo na več nivojih:

- Paralelizem na nivoju ukazov:
  - Nekateri ukazi v programu se lahko izvajajo hkrati - paralelno
  - CPE v obliki **cevovoda**:
    - Izkoriščanje paralelizma na nivoju ukazov
    - ***Pomembna prednost: v programih niso potrebne spremembe !!!***
    - ***Omejen***, iščemo tudi druge možnosti

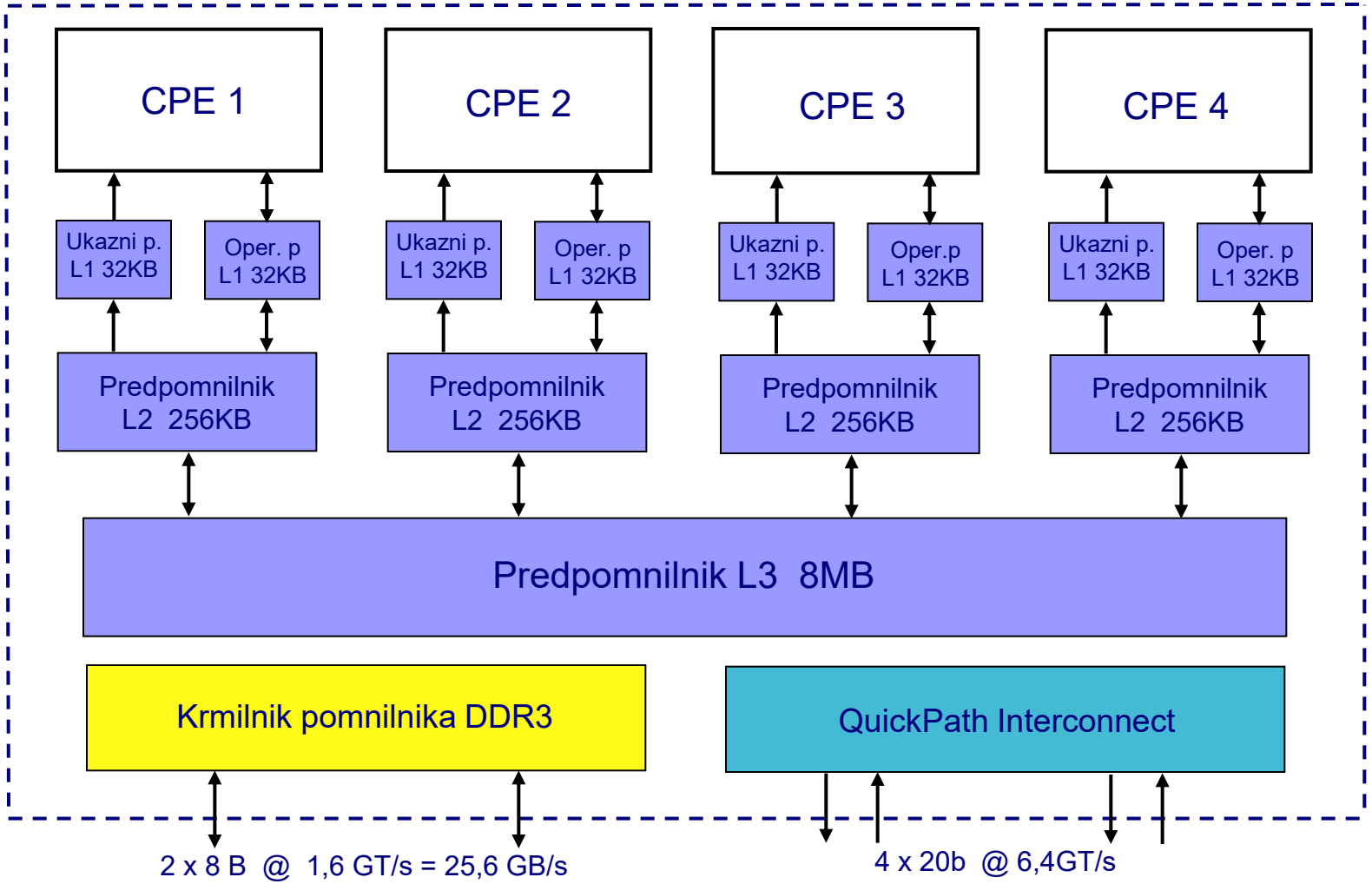
Paralelizmi - vrste

Paralelno izvajanje ukazov

- Paralelizem na nivoju ukazov:
  - Nekateri ukazi v programu se lahko izvajajo hkrati - paralelno
  - CPE v obliki **cevovoda**:
    - Izkoriščanje paralelizma na nivoju ukazov
    - ***Pomembna prednost: v programih niso potrebne spremembe !!!***
    - ***Omejen***, iščemo tudi druge možnosti (višje nivoje)
- Prvi višje-nivojski paralelizem imenujemo **paralelizem na nivoju niti.**
  - Večnitnost
  - Večjedrni procesorji
- Paralelizem na nivoju CPE (MIMD - multiprocesorji, multiračunalniki)
- Paralelizem na nivoju podatkov (GPU, SIMD, Vektorske enote)

- Intel Core i7 Haswell
  - Feature size 22 nm (= 22 * 10^{-9} m)
  - Število tranzistorjev 1,6 milijarde (= 1.600.000.000)
  - Velikost čipa 160 mm^2 (od 10x do 26x mm^2)
  - Frekvenca ure od 2,0 GHz do 4,4 GHz
  - Število jeder (CPE) 4
  - Grafični procesor
  - Podnožje LGA 1150
  - TDP (Thermal Design Power) od 11,5 W do 84 W
  - Cena ≈ 300 – 400 $

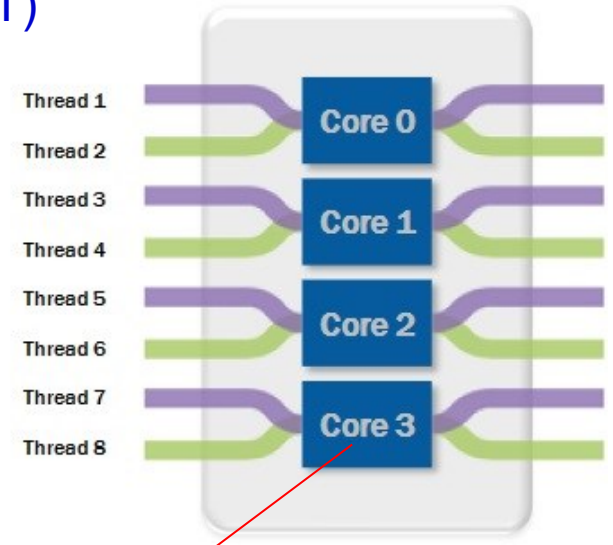
Zgradba 4-jedrnega procesorja Intel Core i7 (Haswell)

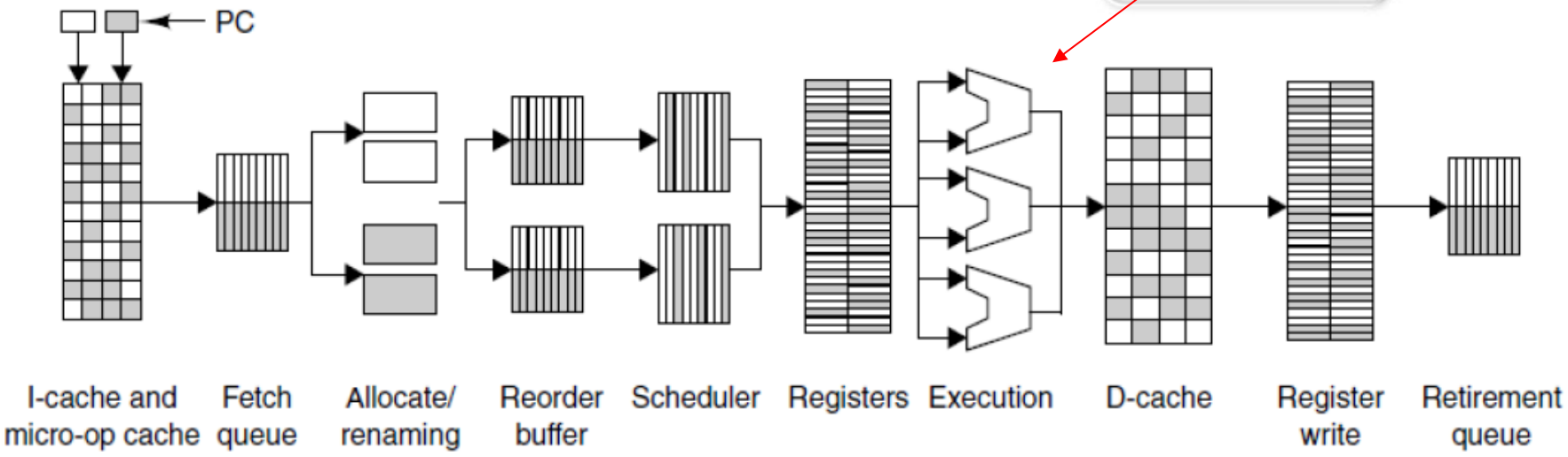


Istočasna večnitnost (SMT)

„Hyperthreading“ na Core i7

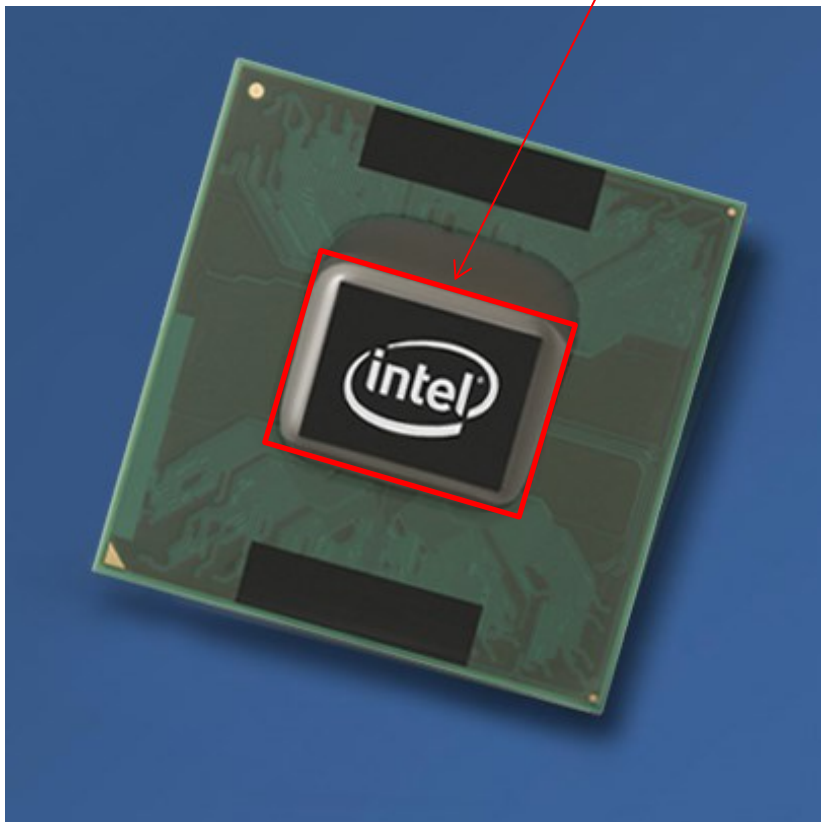
1 jedro podpira 2 niti
(dve „navidezni“ jedri)



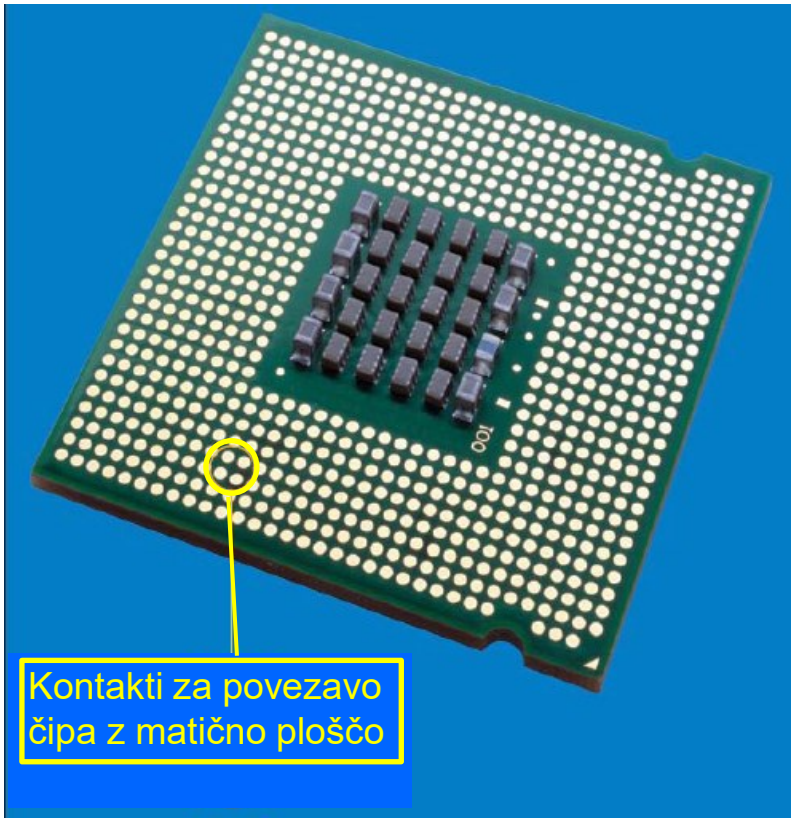


CPE čip na podnožju s kontakti (LGA775)

CPE čip na podnožju s kontakti (LGA775)

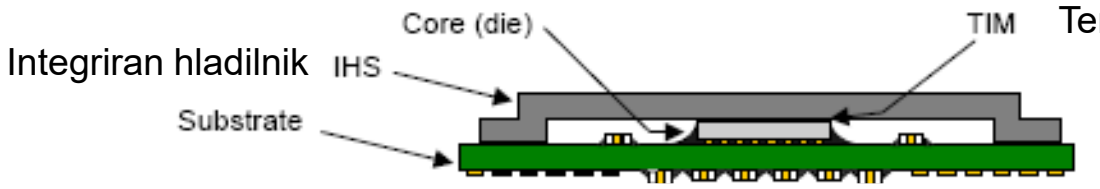


Zgornja stran



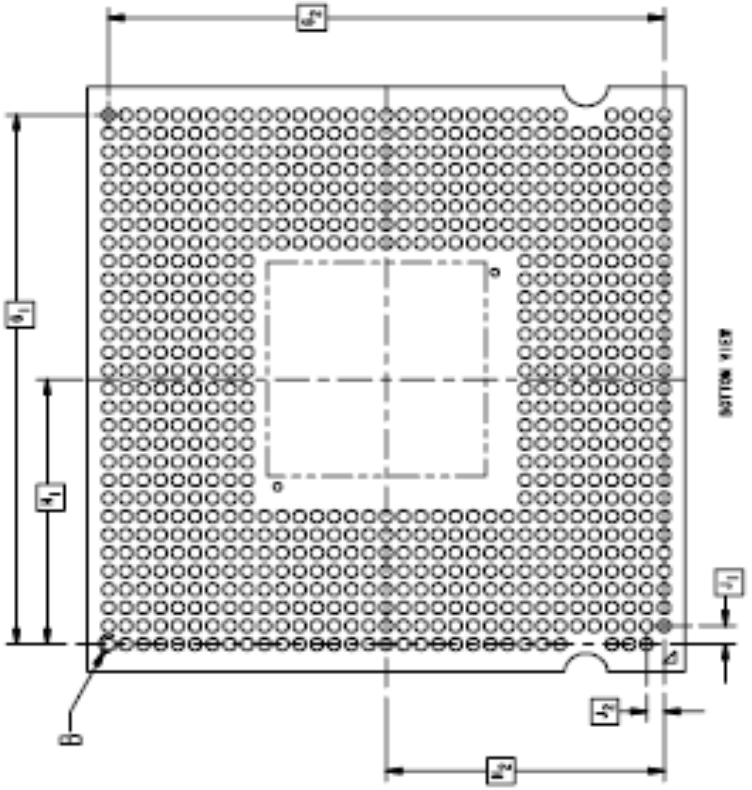
Spodnja stran s kontakti in kondenzatorji

CPE čip na podnožju s kontakti (LGA775) - prerez



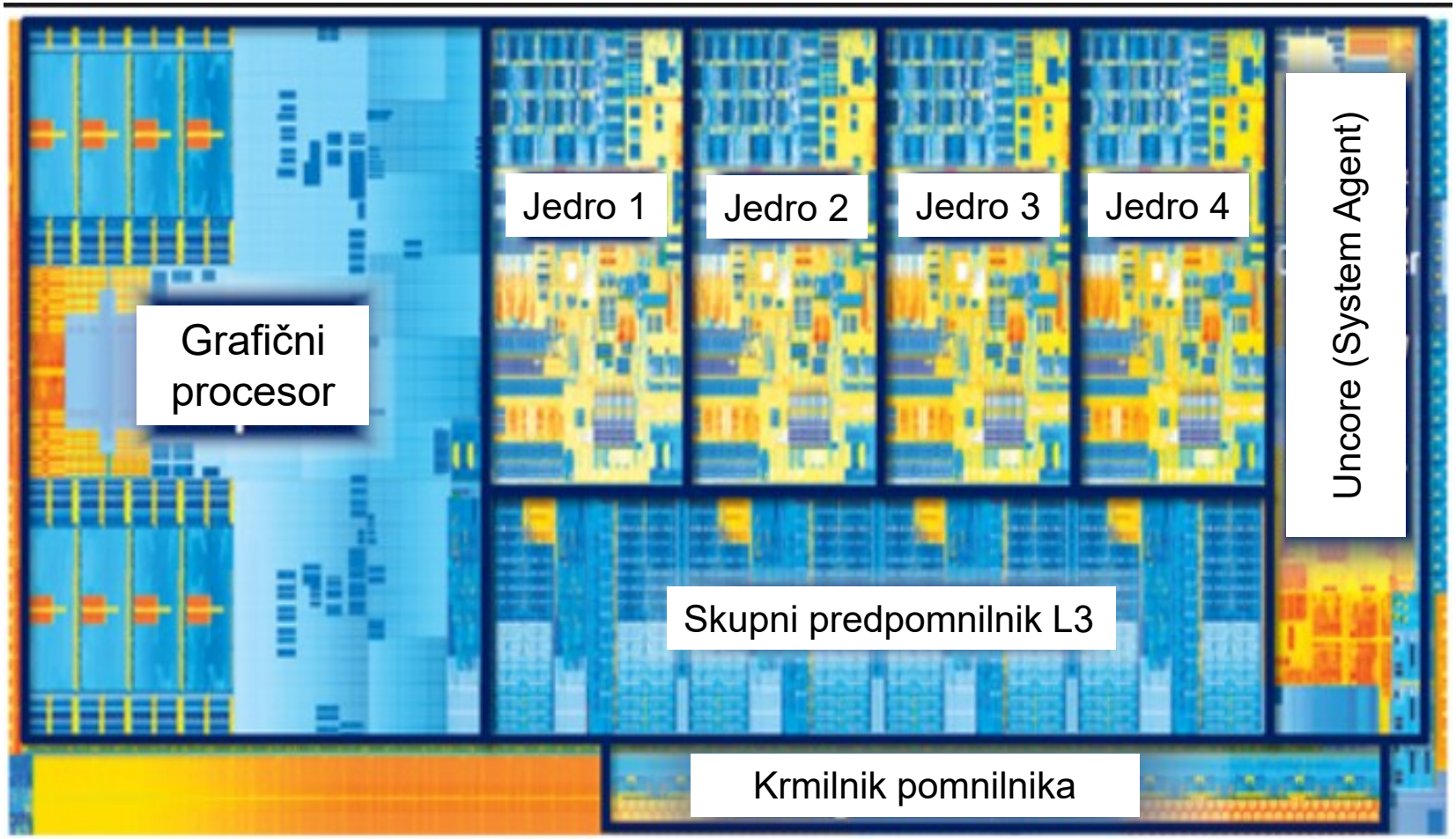
Termično prevoden vmesnik

CPE čip s podnožjem in ohišjem - prerez



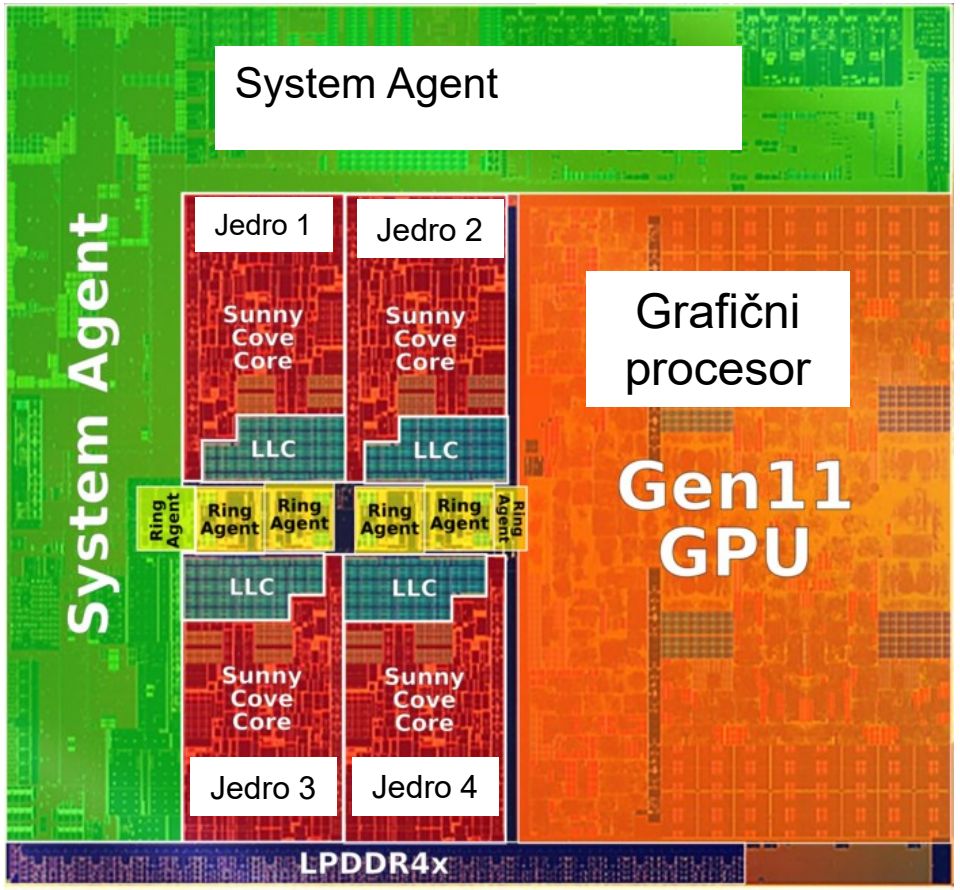
Podnožje LGA775 - pogled s spodnje strani

Čip Intel Core i7 (Haswell)



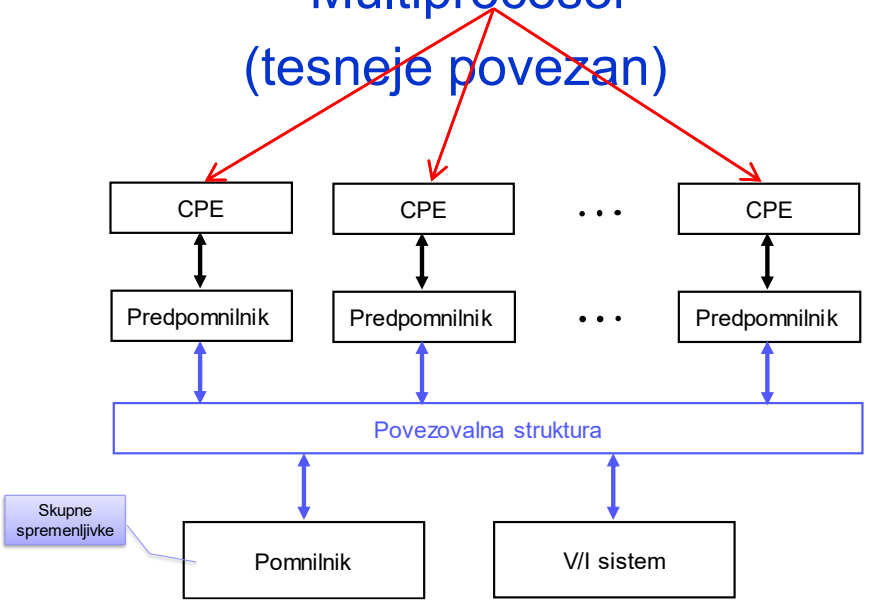
Čip Intel Core i7 (Ice Lake)

(I. 2019)

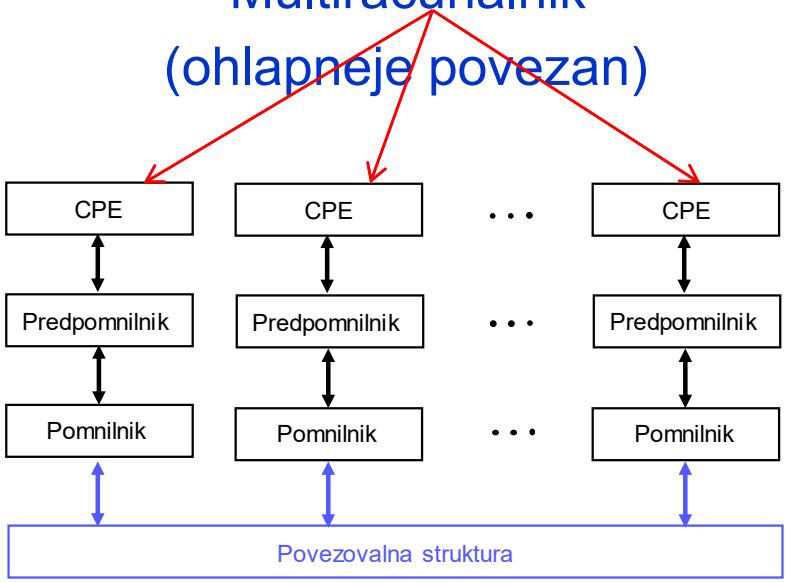


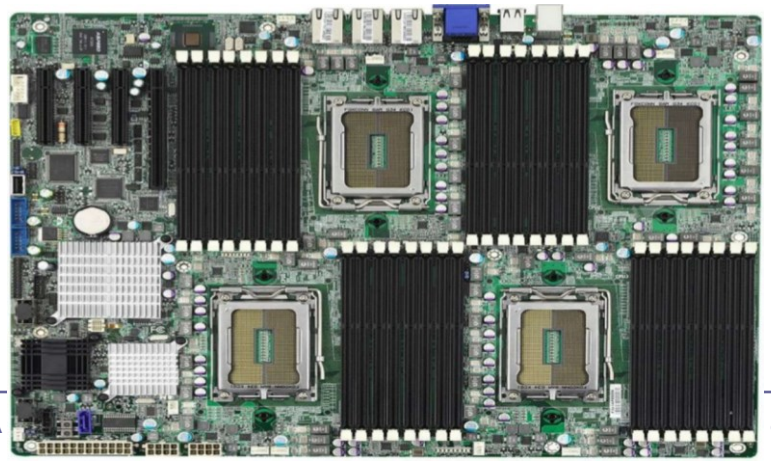
MIMD (Multiple Instruction Multiple Data)

Multiprocesor (tesneje povezan)



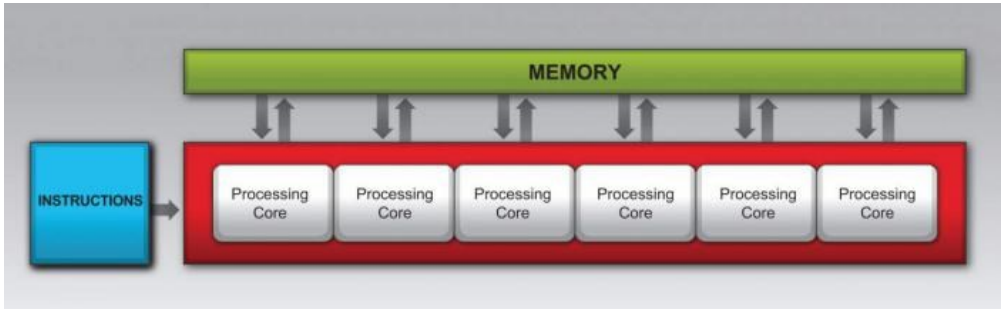
Multiračunalnik (ohlapneje povezan)





Paralelno procesiranje večjega števila podatkov

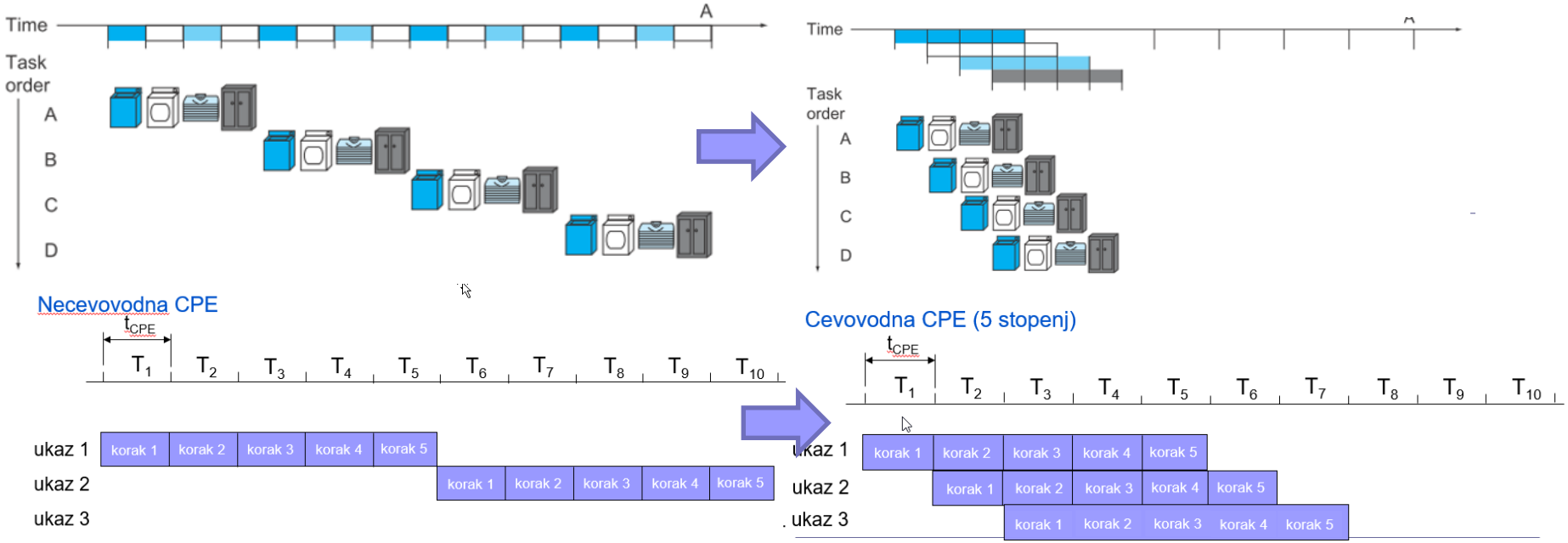
Tesla K40 ima vsega skupaj 1920 procesnih elementov (15 CU * 128 PE v CU).



https://doc.sling.si/workshops/programming-gpu/GPE/teslak40/

6.6 Cevovodna CPE (podatkovna enota)

- Je realizacija CPE, kjer se hkrati izvršuje več ukazov, tako da se posamezni koraki izvrševanja ukazov prekrivajo.
- V cevovodni CPE se ukazi izvršujejo podobno tekočemu traku v proizvodnji (npr. avtomobilov) ali obdelavi oblačil :

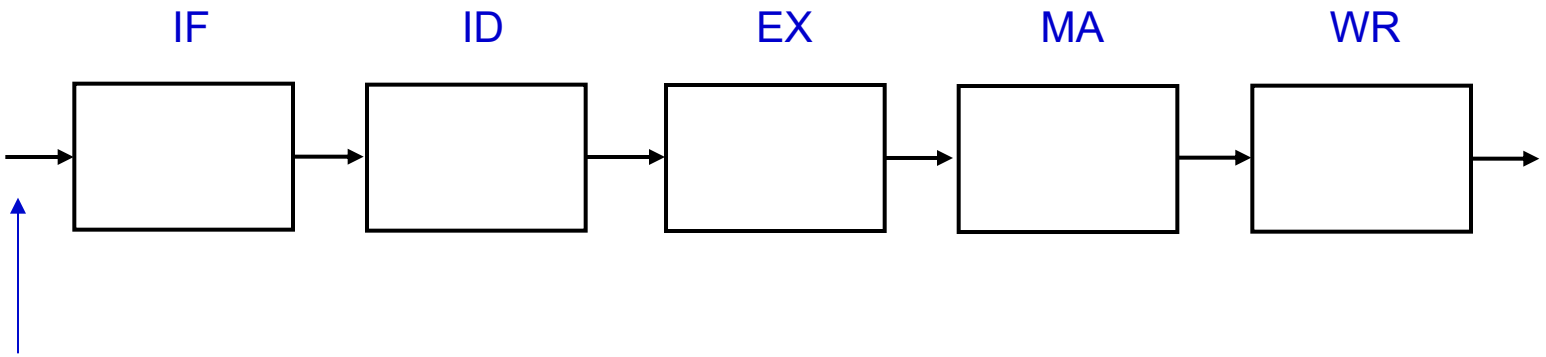


- Izvrševanje ukaza se razdeli na manjše **podoperacije**, za vsako je potreben samo del celotnega časa, ki je potreben za izvršitev ukaza.

Stopnje v cevovodu

- CPE je razdeljena na **stopnje** ali **segmente cevovoda**, ki jih je toliko kot podoperacij v ukazu.
- Vsako podoperacijo izvrši določena stopnja ali segment cevovoda.
- Stopnje so med seboj povezane, ukazi na eni strani vstopajo, potujejo skozi stopnje, v katerih se izvršujejo posamezne podoperacije ukazov in na drugi strani izstopajo.
- V cevovodu je hkrati v izvrševanju toliko ukazov, kot je stopenj.

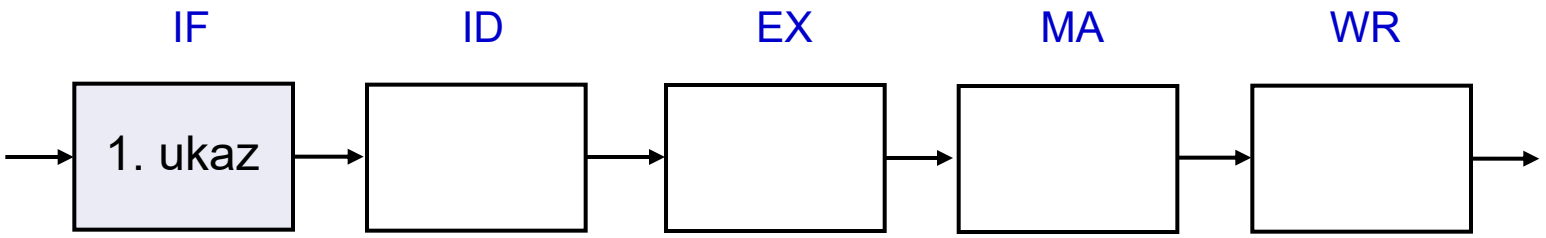
Primer delovanja petstopenjske cevovodne CPE



Na začetku 1. urine periode 1. ukaz vstopi v cevovod

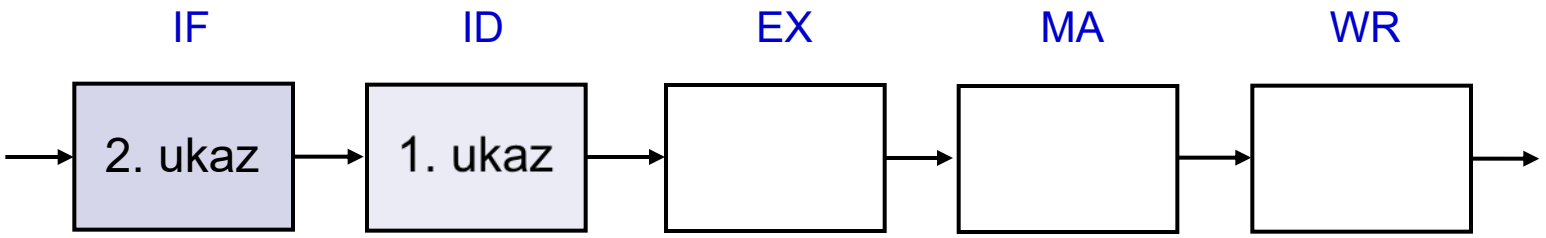
Primer delovanja petstopenjske cevovodne CPE

1. urina perioda



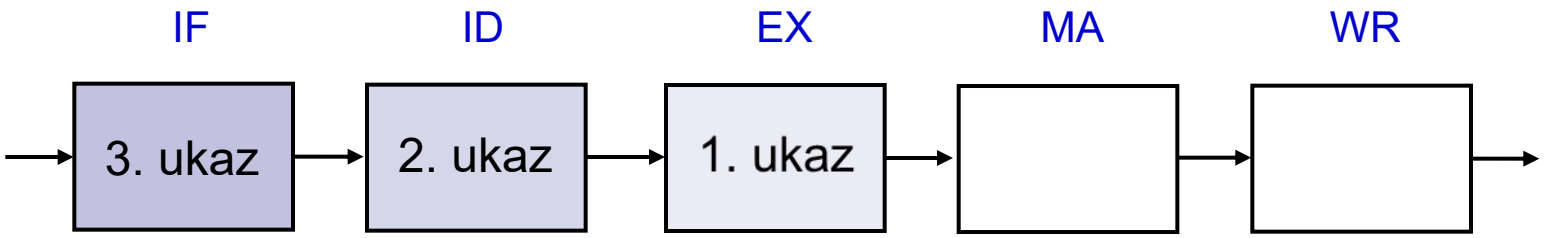
Primer delovanja petstopenjske cevovodne CPE

2. urina perioda



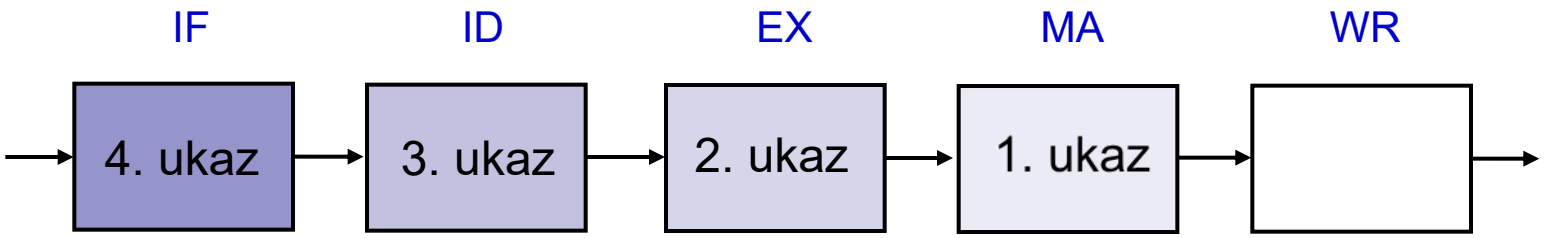
Primer delovanja petstopenjske cevovodne CPE

3. urina perioda



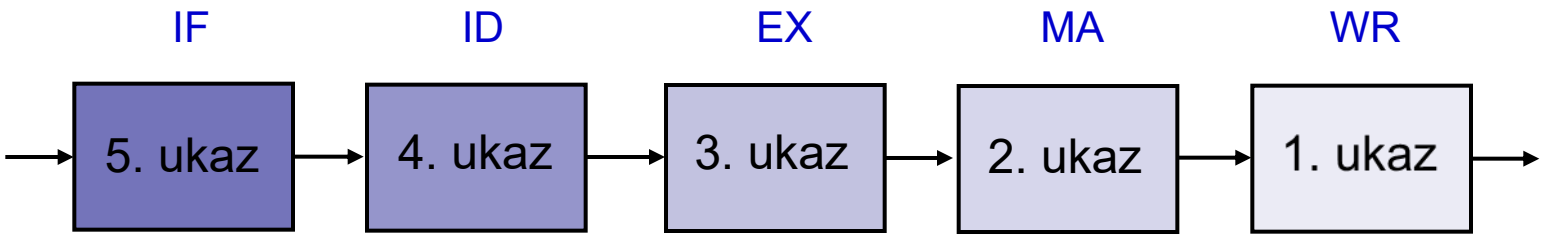
Primer delovanja petstopenjske cevovodne CPE

4. urina perioda



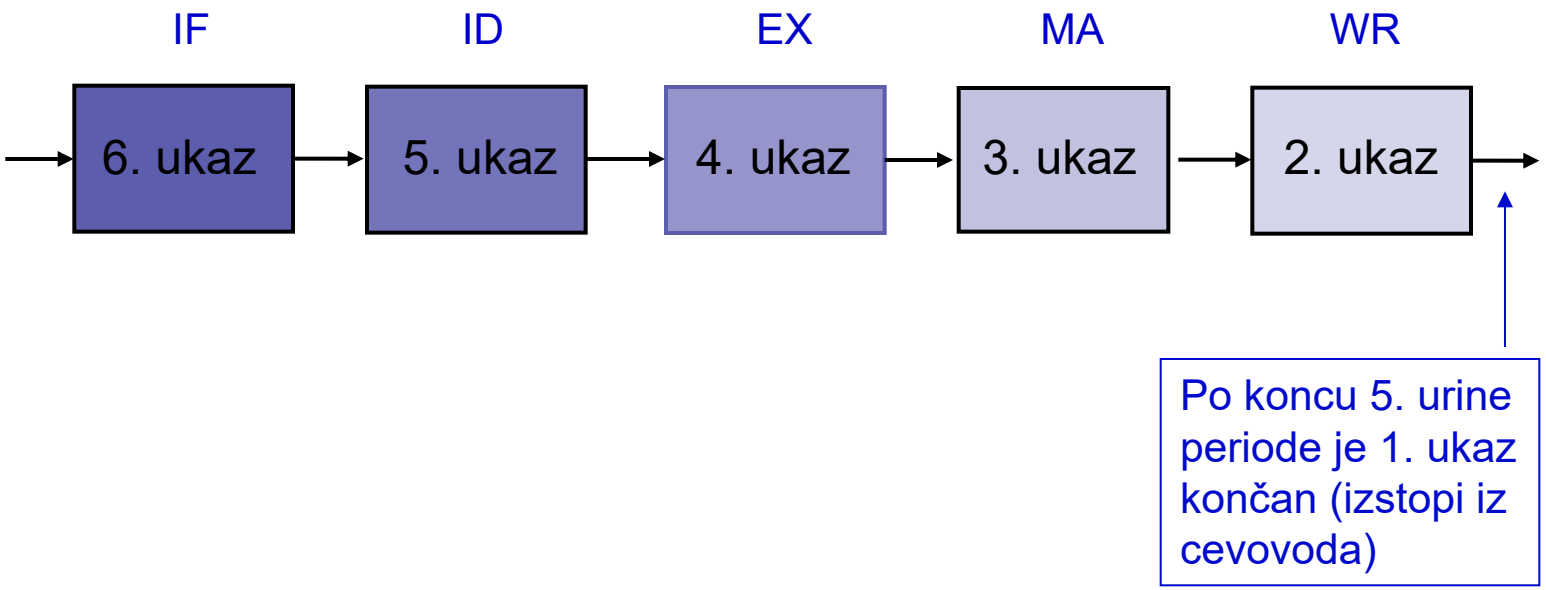
Primer delovanja petstopenjske cevovodne CPE

5. urina perioda

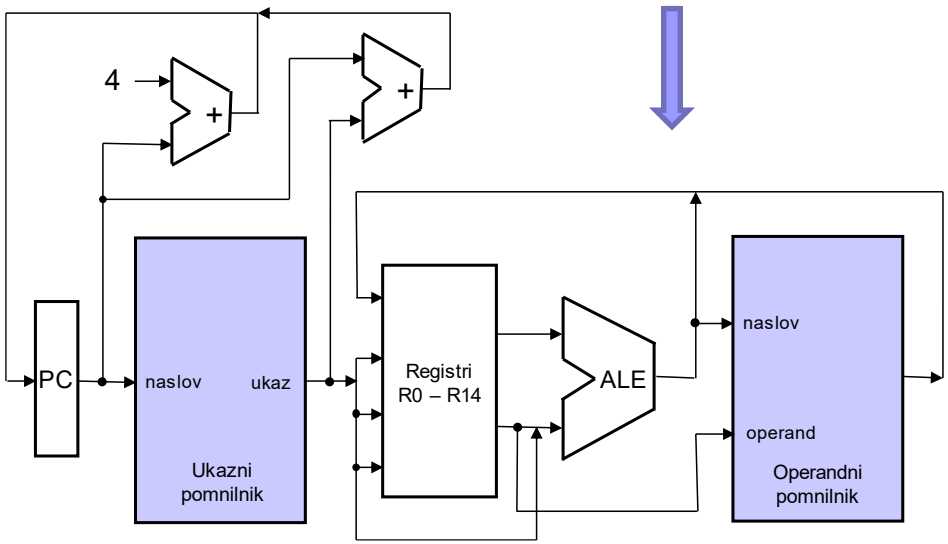


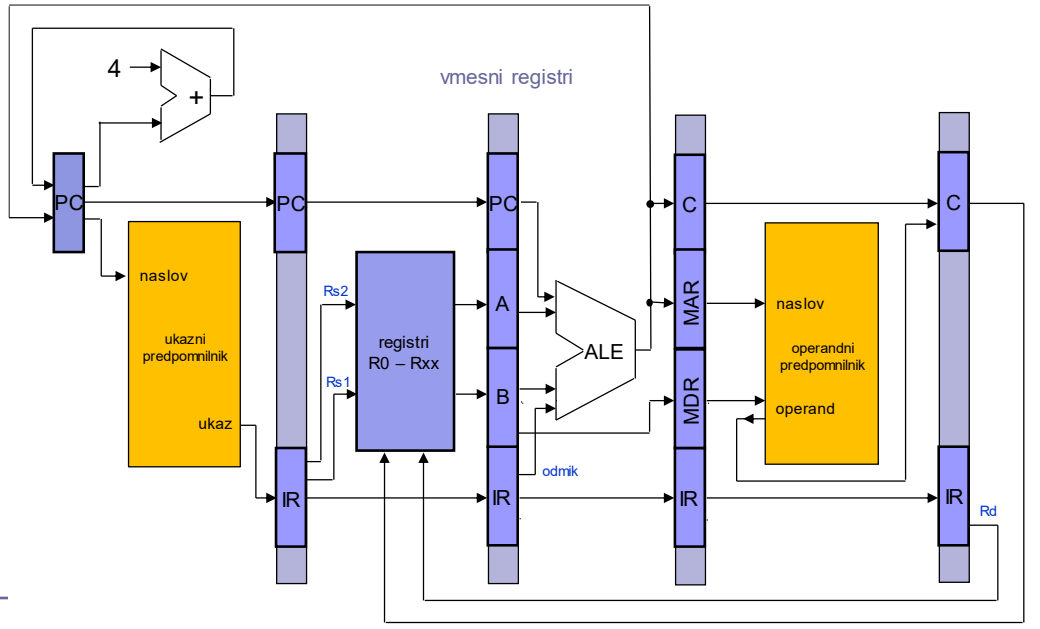
6. urina perioda

Primer delovanja petstopenjske cevovodne CPE



Primerjava necevovodne in 5-stopenjske cevovodne CPE

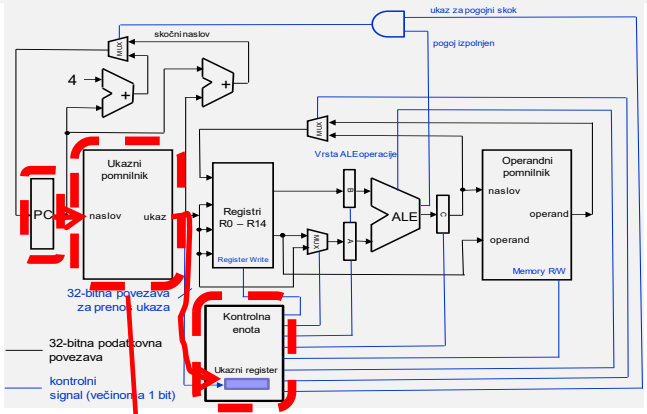




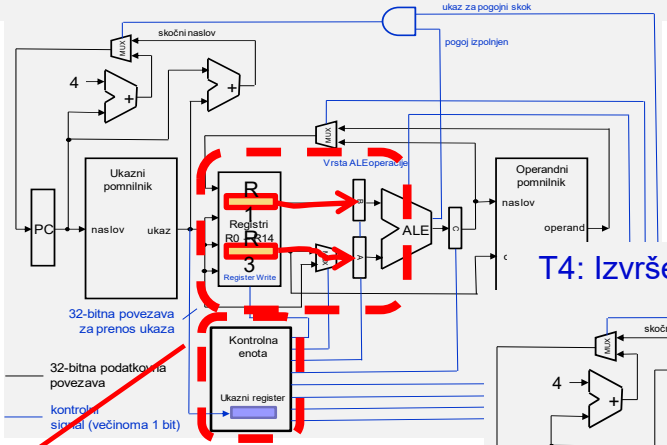
Primerjava izvedbe ukaza v necevovodni in cevovodni CPE

T1: Dostop do ukaza v ukaznem pomnilniku

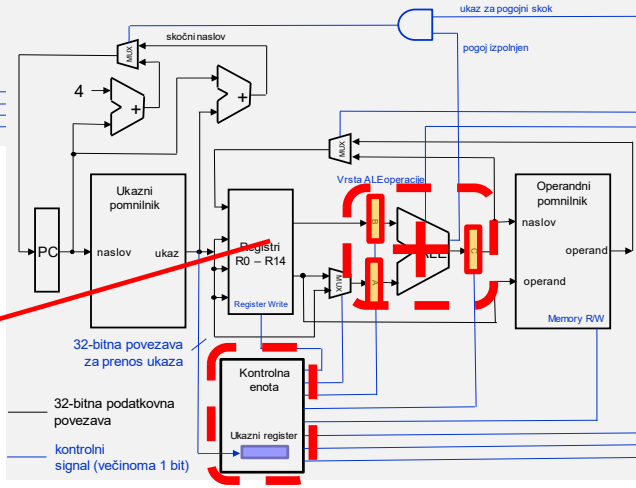
T2: Prenos ukaza iz pomnilnika v ukazni register



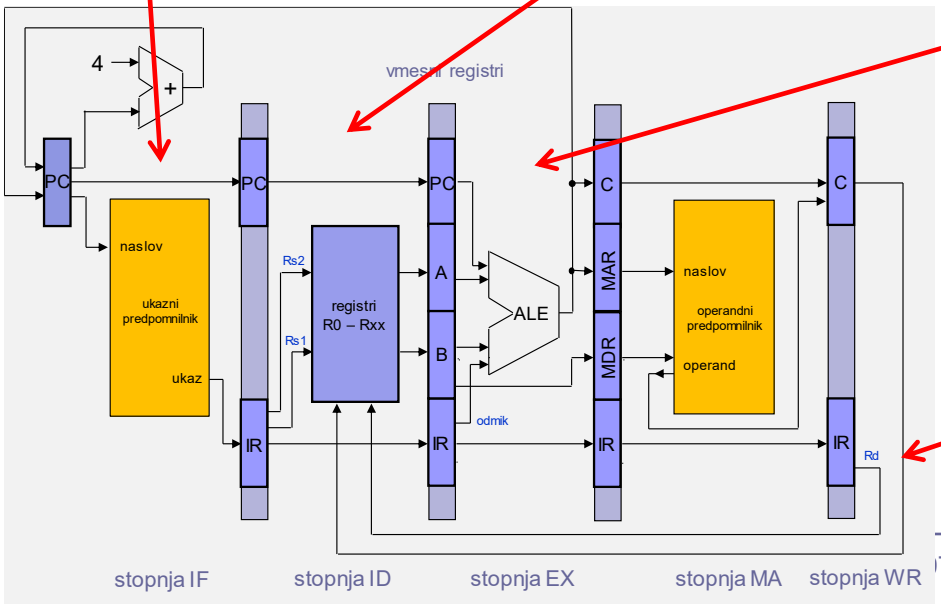
T3: Dekodiranje ukaza in dostop do operandov v R1 in R3



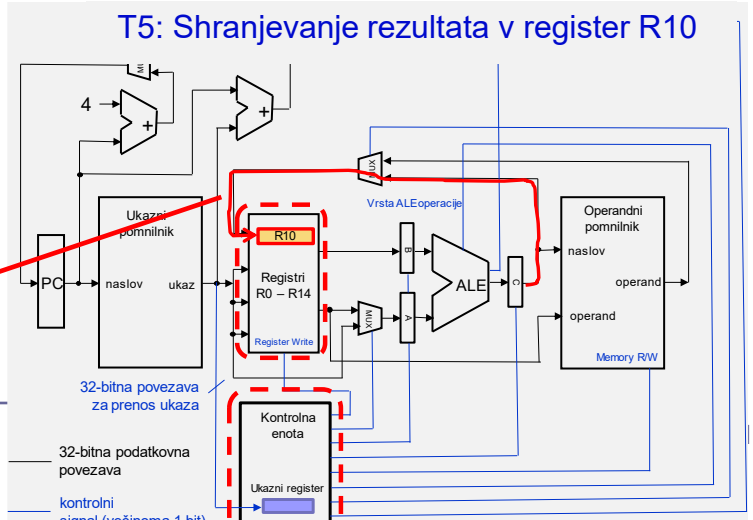
T4: Izvrševanje operacije (seštevanje)



ADD R10, R1, R3



T5: Shranjevanje rezultata v register R10



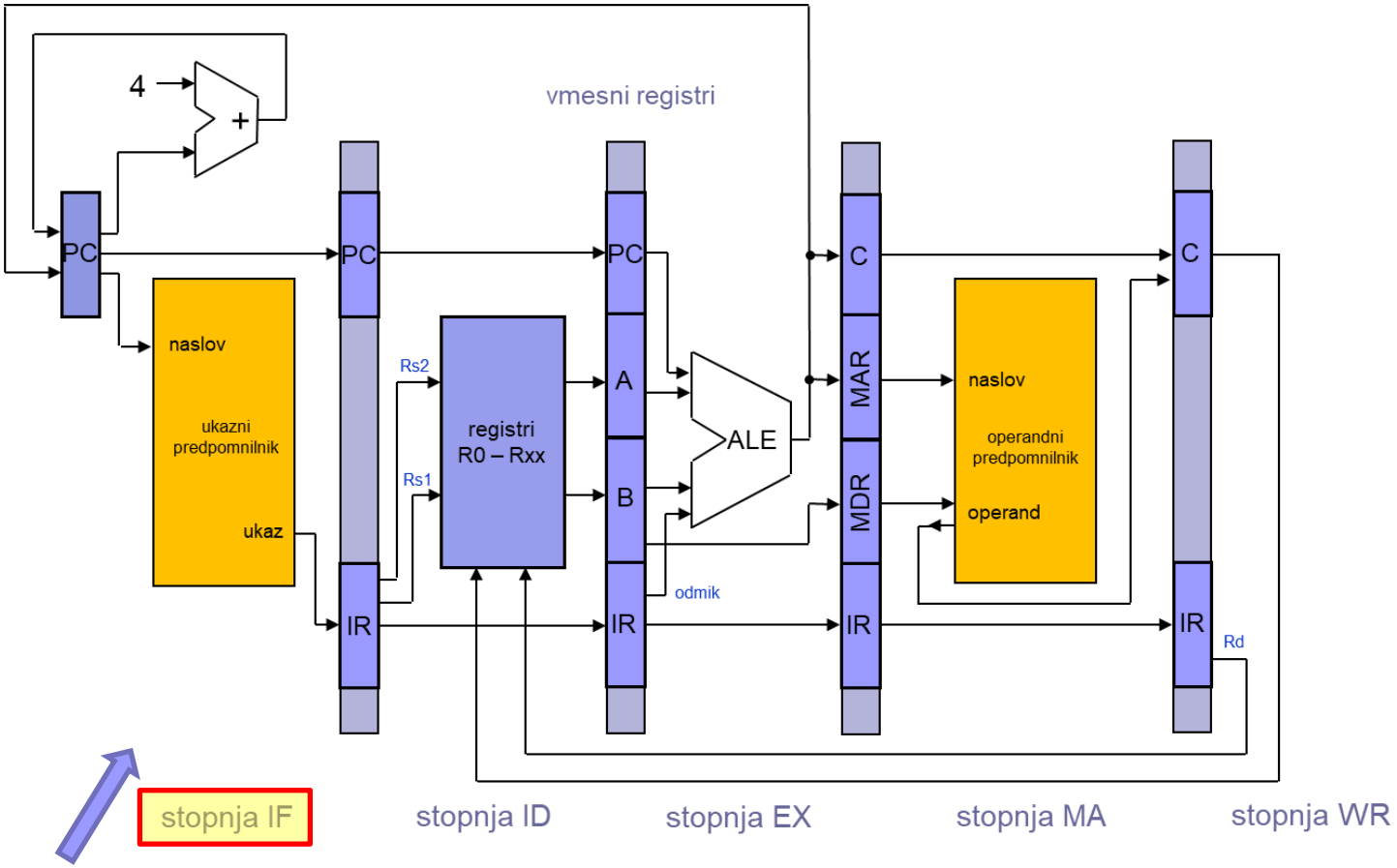
Izvedba ukazov v 5 stopnjah

- Izvajanje ukazov lahko razdelimo na primer na 5 splošnih elementarnih korakov (5 stopenjski cevovod) :
  - Prevzem ukaza (IF - Instruction Fetch)
  - Dekodiranje ukaza in dostop do registrov (ID - Instruction Decode)
  - Izvrševanje ukaza (EX - Execute)
  - Dostop do pomnilnika (MA - Memory Access)
    - (samo pri ukazih LOAD in STORE)
  - Shranjevanje rezultata v register (WR - Write Register)
- Če lahko vse ukaze poenotimo v te elementarne korake (so dovolj enostavni), potem lahko izvajanje ukazov tudi pohitrimo:
  - izvajamo jih več hkrati (vsakega v svojem elementarnem koraku)

    -> cevovod

- Zmogljivost cevovodne CPE je določena s hitrostjo izstopanja ukazov iz cevovoda.
- Ker so stopnje med seboj povezane, se mora premik ukaza iz ene stopnje v drugo izvršiti pri vseh hkrati.
- Premik se običajno izvede vsako urino periodo.
- Trajanje urine periode t_{CPE} zato ne more biti krajše kot je čas, ki ga potrebuje za izvedbo podoperacije najpočasnejša stopnja cevovoda.

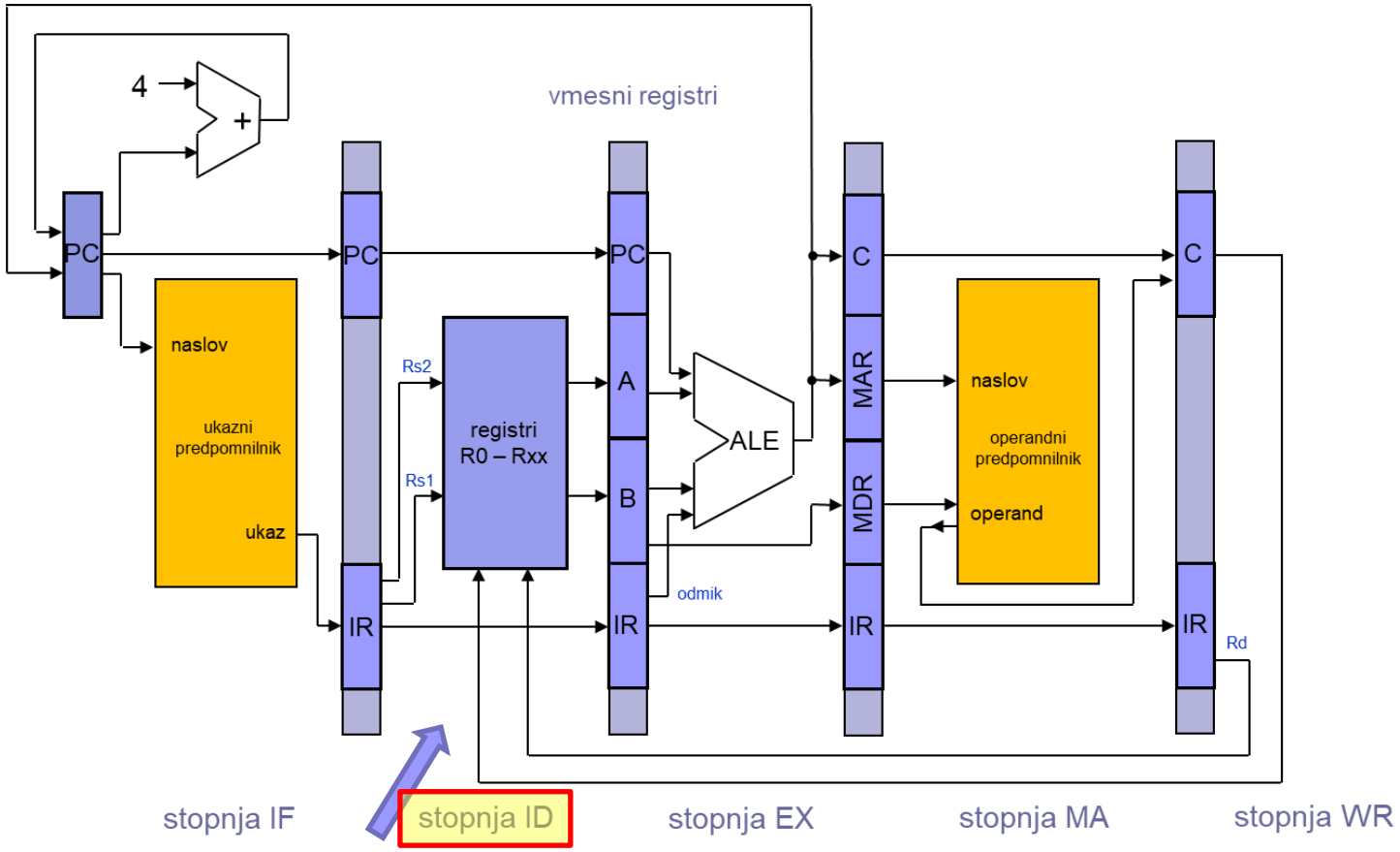
Primer petstopenjske cevovodne CPE - IF



IF = Instruction Fetch
Prevzem ukaza

1. urina perioda

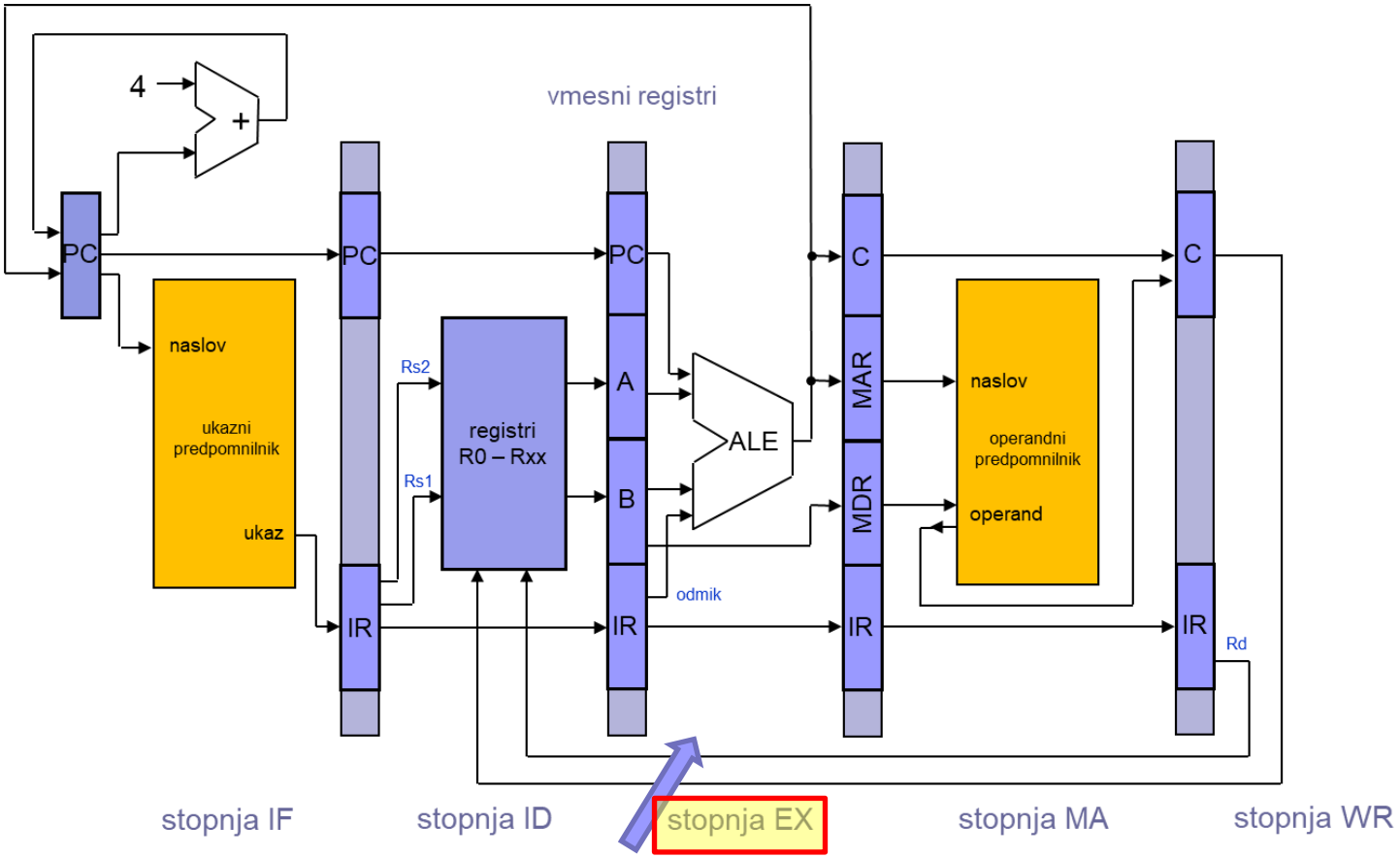
Primer petstopenjske cevovodne CPE - ID



ID = Instruction Decode
Dekodiranje ukaza in dostop do operandov v registrih

2. urina perioda

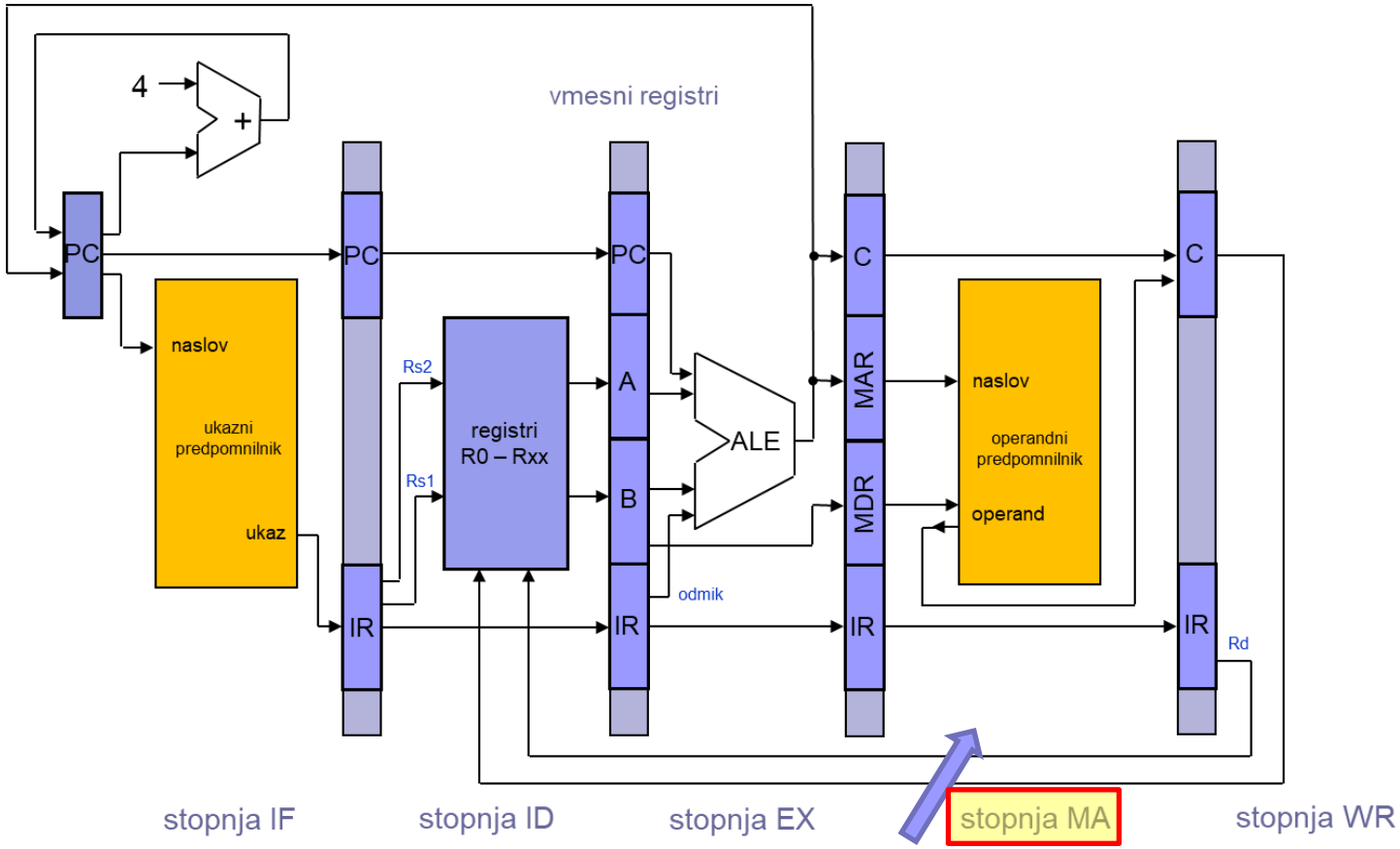
Primer petstopenjske cevovodne CPE - EX



EX = execute
Izvrševanje operacije

3. urina perioda

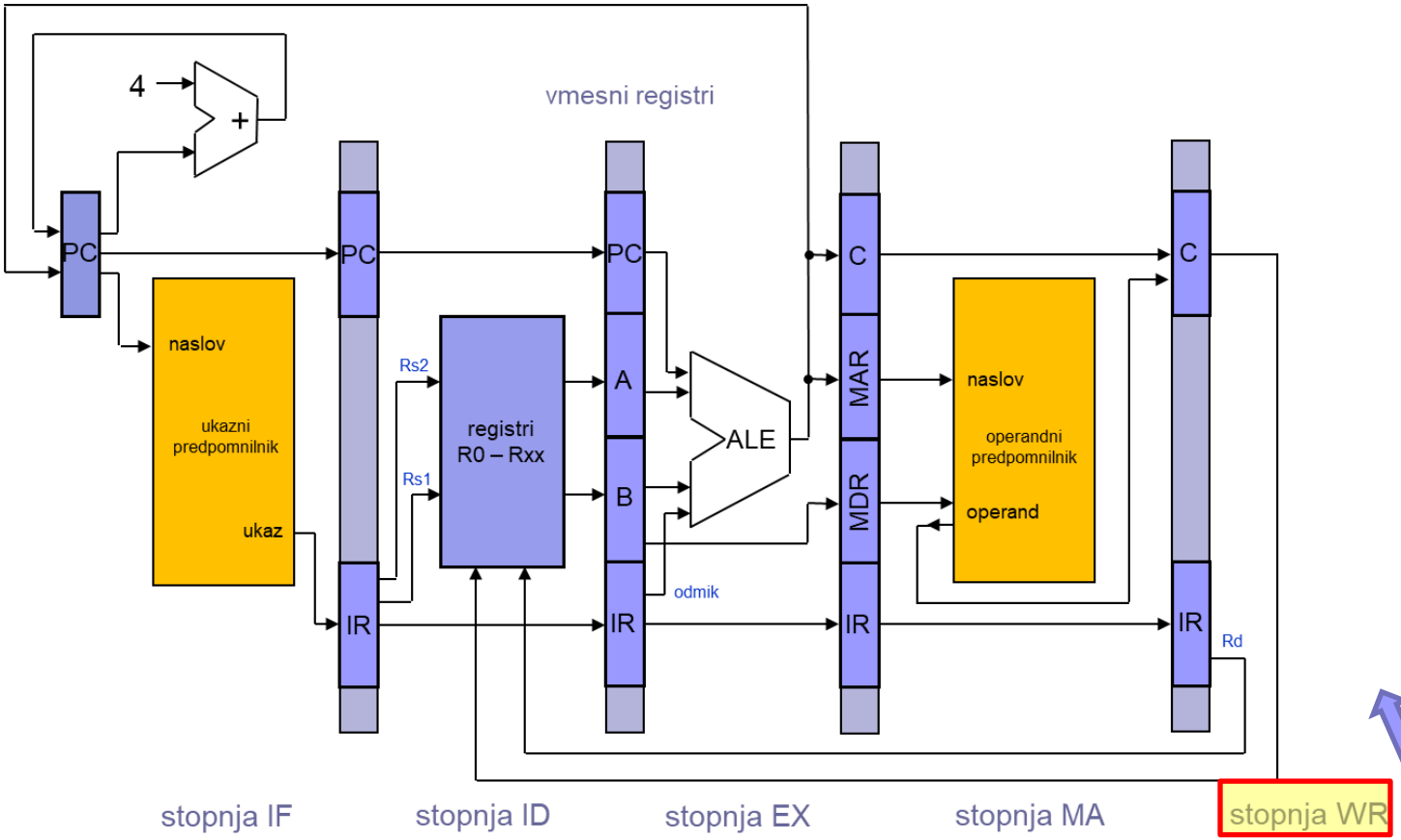
Primer petstopenjske cevovodne CPE - MA



Dostop do operandov
v pomnilniku (LOAD/STORE)
MA = Memory Access

4. urina perioda

Primer petstopenjske cevovodne CPE – WR (WB)

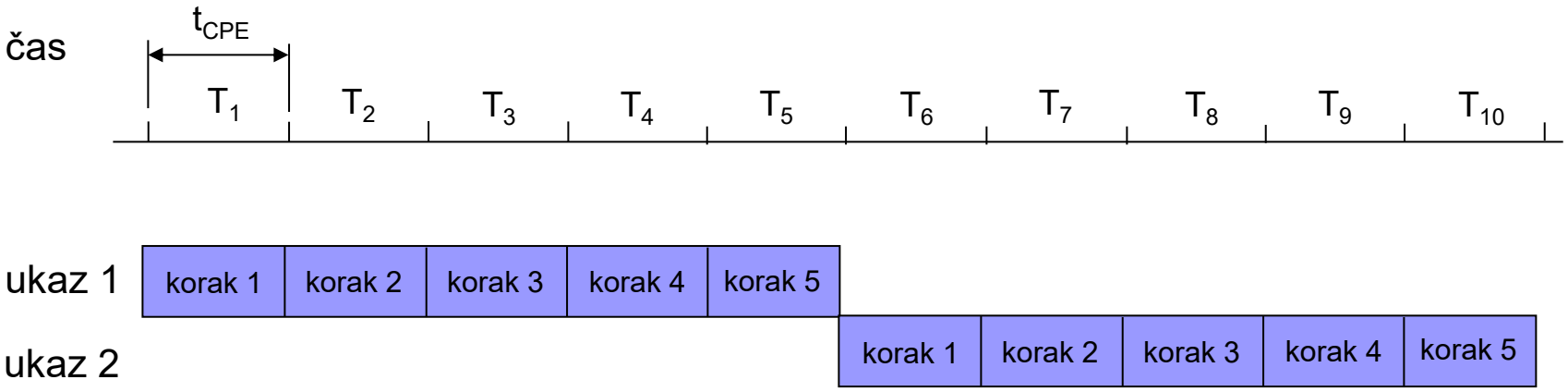


WR = Write Register
Shranjevanje rezultata (v register)

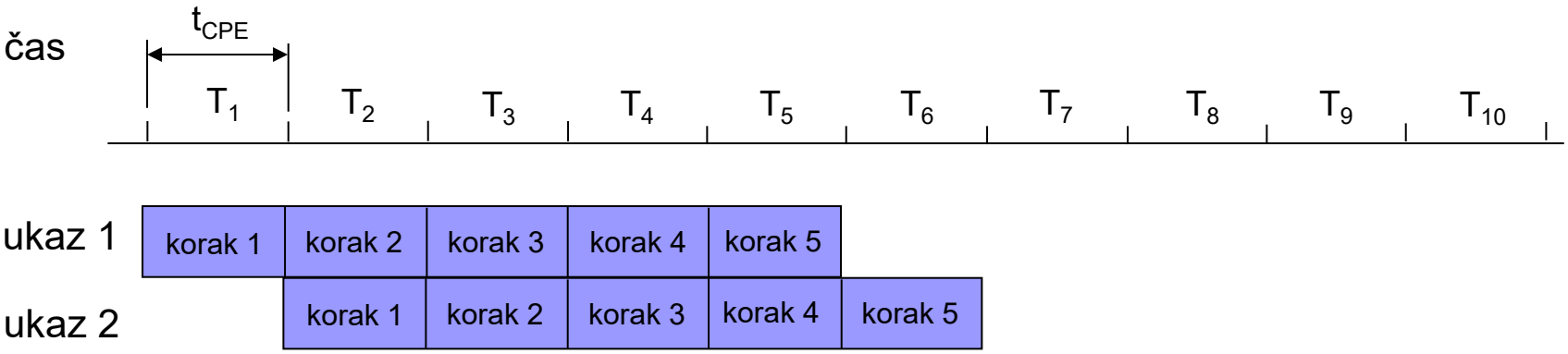
5. urina perioda

Izvrševanje ukazov pri necevovodni in cevovodni CPE

Necevovodna CPE



Cevovodna CPE



Primerjava: koliko urinih period traja izvajanje treh (ali 100) ukazov v obeh primerih ?

- Danes so vsi zmogljivejši procesorji narejeni kot cevovodni procesorji.
- Pri razvoju cevovodne CPE se stremi za tem, da izvrševanje vseh podoperacij traja približno enako dolgo - uravnotežen cevovod.
- Pri idealno uravnoteženi CPE z *N* stopnjami ali segmenti je zmogljivost *N*-krat večja kot pri necevovodni CPE.
- Vsak posamezen ukaz se ne izvrši nič hitreje, se pa v cevovodu hkrati izvršuje *N* ukazov.

- Na izhodu cevovoda dobimo v enakem času *N*-krat več izvršenih ukazov kot pri necevovodni CPE.

- Povprečno število urinih period na ukaz (*CPI*) je v idealnem primeru *N*-krat manjše kot pri necevovodni CPE.

- Trajanje izvrševanja posameznega ukaza (latenca) pa je enako $N \times t_{CPE}$, torej pri enaki urini periodi enako kot pri necevovodni CPE.

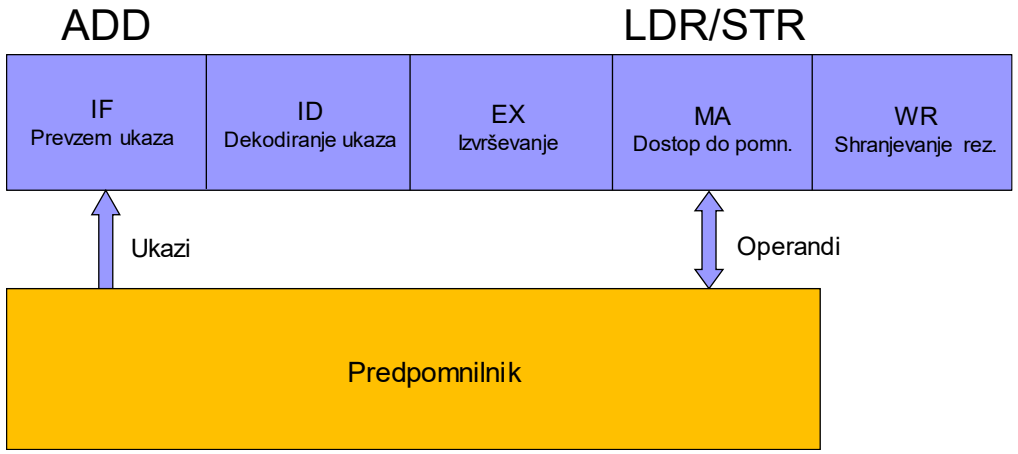
- Ali bi pri dovolj velikem številu stopenj *N* lahko naredili poljubno hitro CPE (*N-krat* hitrejšo)?
  - Ne. Ukazi, ki so istočasno v cevovodu (vsak v svoji stopnji), so lahko med seboj na nek način odvisni in se zato določen ukaz ne more vedno izvajati v naslednji urini periodi.
- Te dogodke imenujemo **cevovodne nevarnosti** (pipeline hazards).

Cevovodne nevarnosti

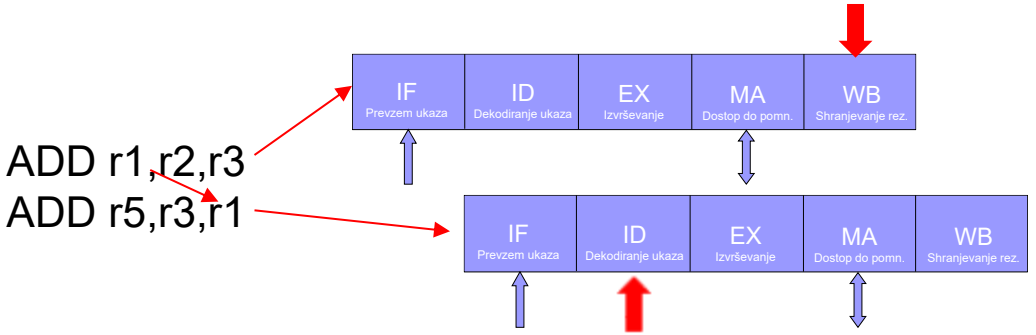
- Razlikujemo tri vrste cevovodnih nevarnosti:
  - **Strukturne nevarnosti** - kadar več stopenj cevovoda v isti urini periodi potrebuje isto enoto
  - **Podatkovne nevarnosti** - kadar nek ukaz potrebuje rezultat predhodnega ukaza, ki pa še ni končan
  - **Kontrolne nevarnosti** - pri ukazih, ki spreminjajo vrednost PC (kontrolni ukazi: skoki, klici, …)

Vrste cevovodnih nevarnosti

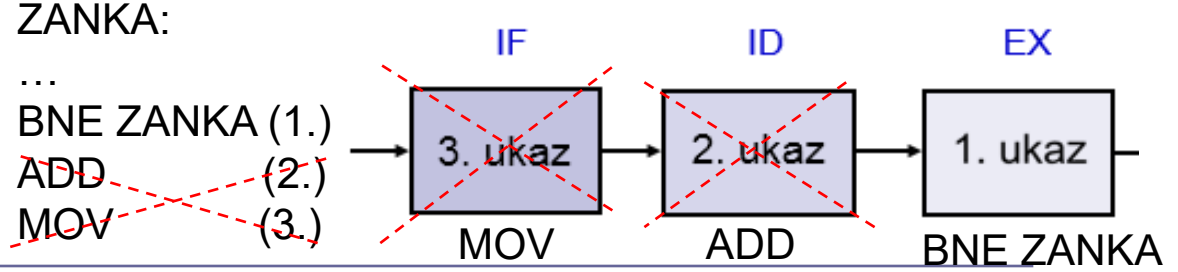
- Strukturne nevarnosti
  - Dostop do iste enote (npr. predpomn.)



- Podatkovne nevarnosti
  - Medsebojna odvisnost med ukazi



- Kontrolne nevarnosti
  - Skočni ukazi (polnjenje cevovoda)



- Zaradi cevovodnih nevarnosti moramo poskrbeti za pravilno izvajanje programa, zato :
  - se mora vsaj del cevovoda ustaviti dokler nevarnost ne mine (cevovod takrat ne sprejema novih ukazov – „zaklenitev“)
    - Manj učinkovita rešitev !

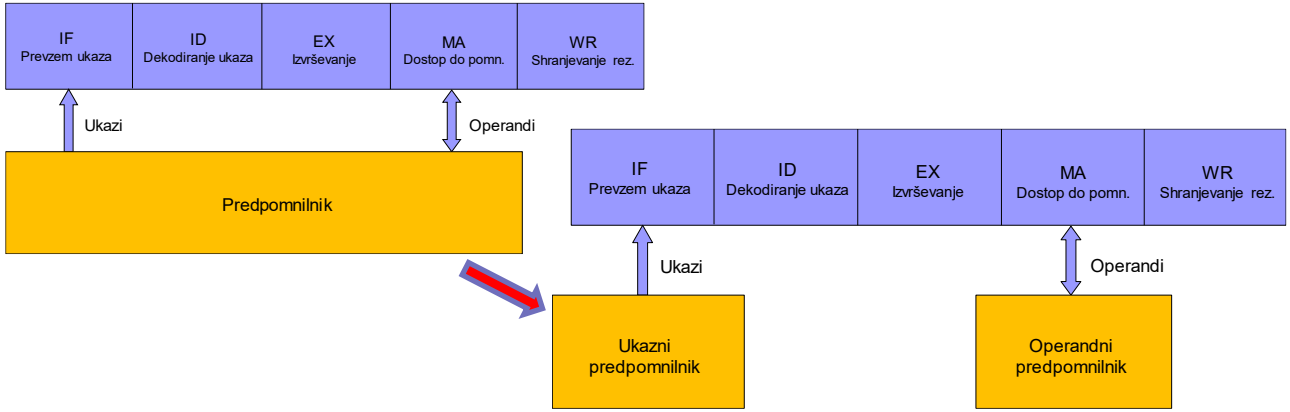
- Zaradi cevovodnih nevarnosti :
  - Povečanje hitrosti **ni *N-kratno***.
  - Realni CPI se poveča: CPI > 1 (Idealni cevovodni CPI = 1)

- Iščemo bolj učinkovite rešitve za cevovodne nevarnosti

Cevovodne nevarnosti - rešitve

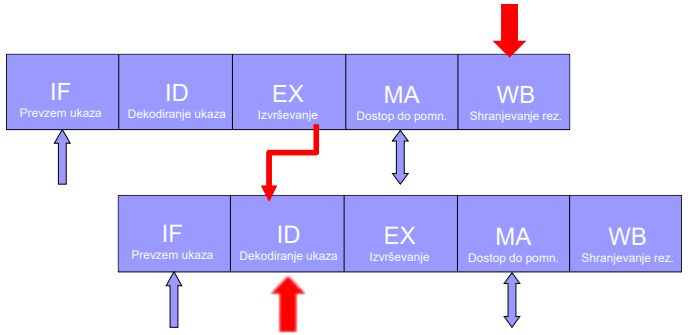
Cevovodna CPE – primeri **bolj učinkovitih rešitev** cevovodnih nevarnosti:

- Strukturne nevarnosti
  - Rešitev –> ločitev predpomnilnikov
    - ukazi, operandi
    - Harvardska arh.



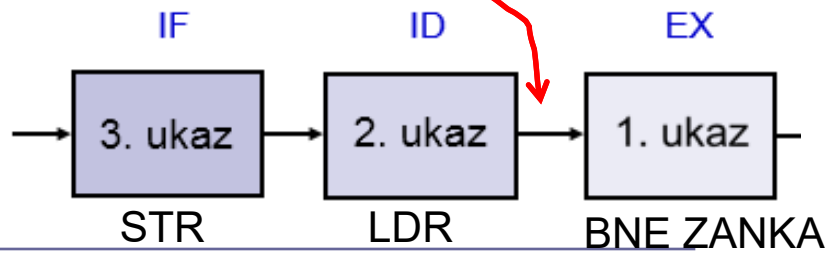
- Podatkovne nevarnosti
  - Pomaga tudi razvrščanje ukazov (program)
  - Rešitev -> premoščanje

ADD r1,r2,r3
ADD r5,r3,r1



- Kontrolne nevarnosti
  - Rešitev -> napoved skoka (pogoja in naslova)
    - Polnimo cevovod s „pravimi“ ukazi

ZANKA:
LDR (2.)
STR (3.)
BNE ZANKA (1.)
ADD
MOV



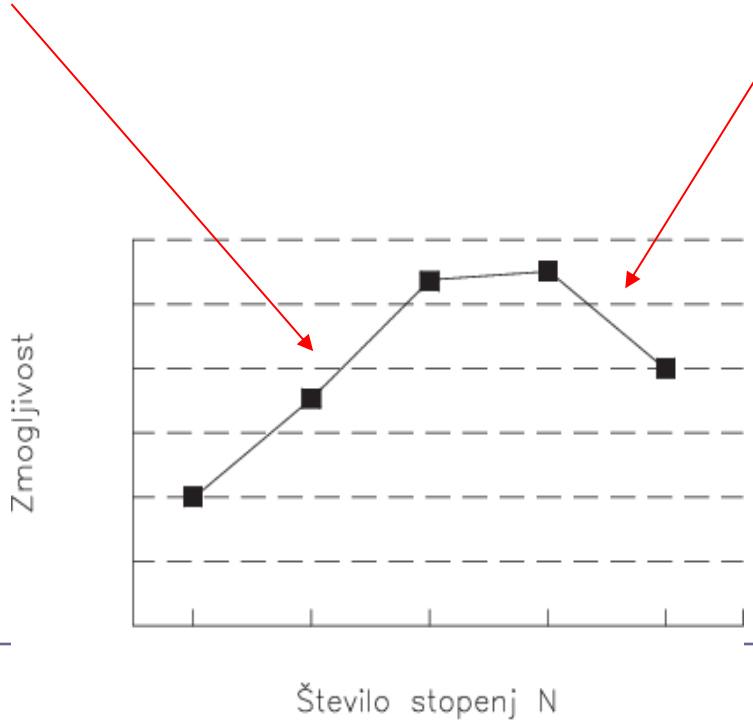
Cevovodna CPE – število stopenj

Ideal:

- N stopenj – N-kratna pohitritev ?
- Idealni primer: CPI = 1

Realnost:

- Ali bi pri dovolj velikem številu stopenj *N* lahko naredili poljubno hitro CPE?
  - Ne!
- Daljši kot je cevovod, več je ukazov hkrati v cevovodu in cevovodne nevarnosti se pojavljajo pogosteje.



6.7 Primeri cevovodnih CPE

- Poenostavljen 5-stopenjski cevovod
- FRI SMS Atmel 9260 ARMv5

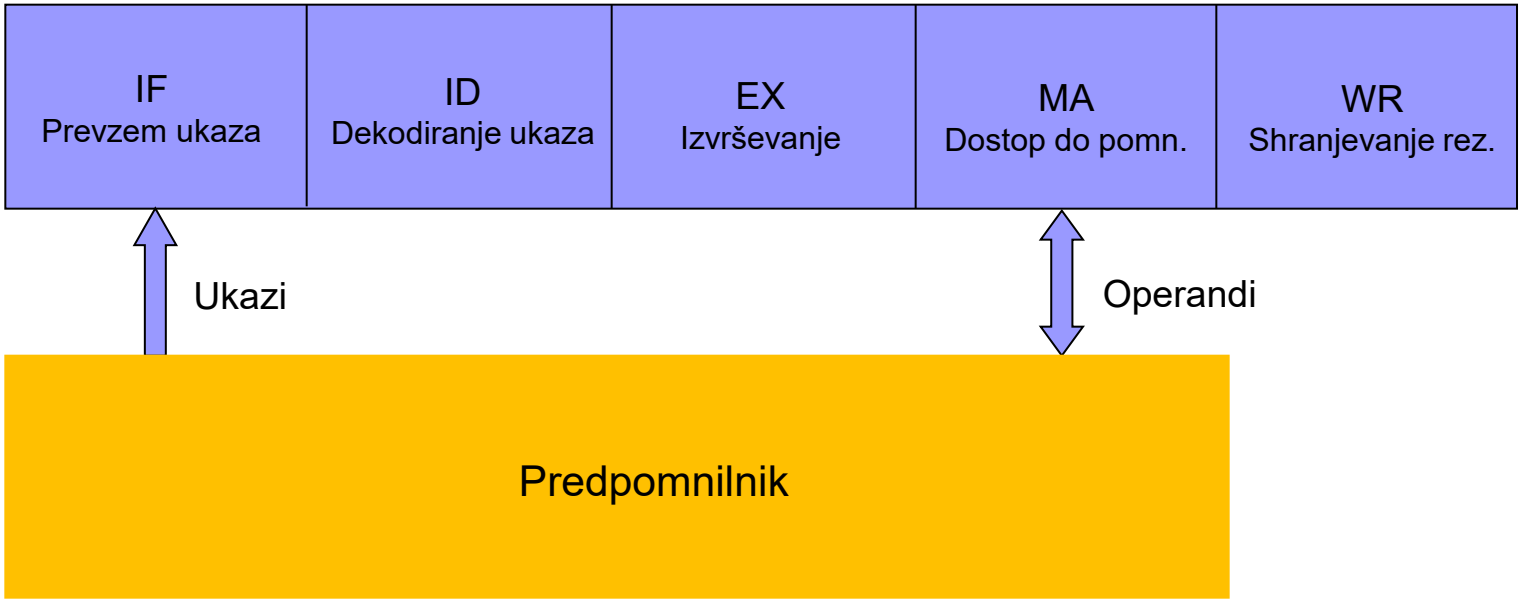
Poenostavljen 5-stopenjski cevovod

- Osnova naj bo izvajanje ukazov v petih korakih, kot smo ga opisali v prejšnjem poglavju.
- Izvrševanje ukaza razdelimo na pet podoperacij v skladu s koraki iz prejšnjega poglavja, CPE pa v pet stopenj oziroma segmentov:
  - Stopnja IF (Instruction Fetch) - prevzem ukaza
  - Stopnja ID (Instruction Decode) - dekodiranje ukaza in dostop do registrov
  - Stopnja EX (Execute) - izvrševanje operacije
  - Stopnja MA (Memory Access) - dostop do pomnilnika
  - Stopnja WR (Write Register) - shranjevanje rezultata

- Vsaka stopnja cevovoda mora izvršiti svojo podoperacijo v eni urini periodi.

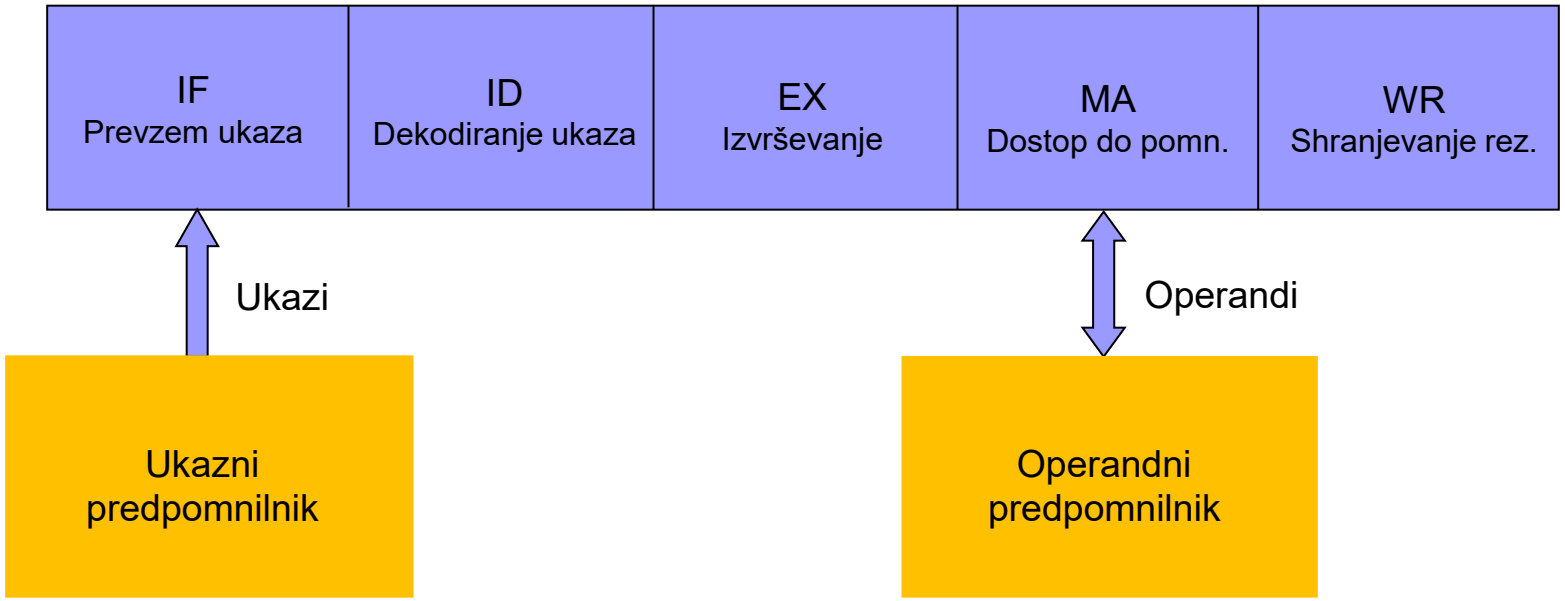
- V stopnjah IF in MA lahko pride do hkratnega dostopa do pomnilnika (v isti urini periodi) - strukturna nevarnost.

- Da odpravimo to vrsto strukturne nevarnosti, moramo predpomnilnik razdeliti v ukazni in operandni predpomnilnik (Harvardska arhitektura).



Pri hkratnem dostopu do ukaza (stopnja IF) in operanda v predpomnilniku (stopnja MA), pride do strukturne cevovodne nevarnosti

Cevovodna CPE – razrešitev strukturne nevarnosti dostopa do predpomnilnika



Strukturna nevarnost, ki bi nastopila zaradi hkratnega dostopa stopenj IF in MA do pomnilnika, je odpravljena s Harvardsko arhitekturo predpomnilnika

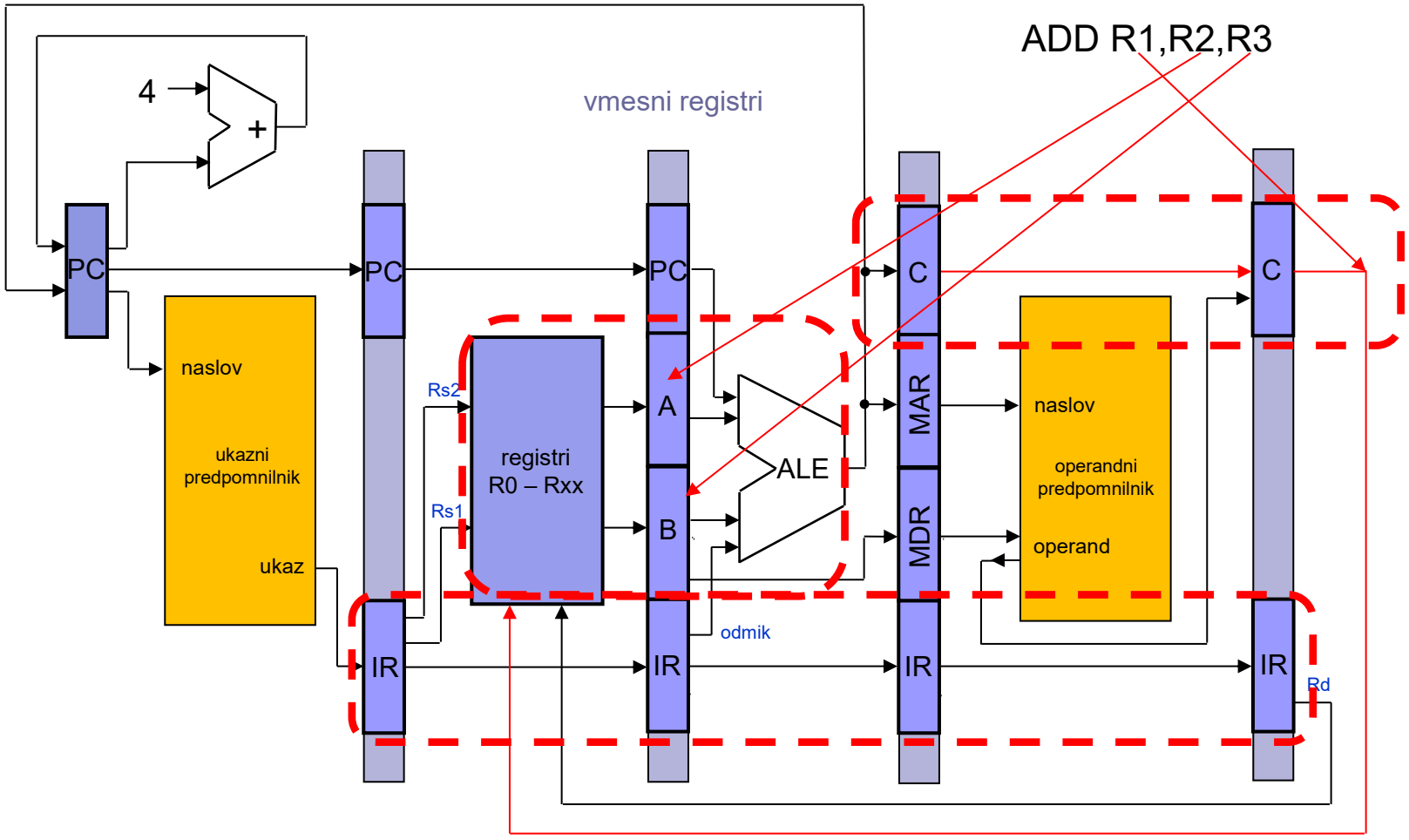
- V stopnji IF se dostop do ukaznega predpomnilnika opravi vsako urino periodo, pri necevovodni CPE pa (v našem primeru) samo na vsakih pet urinih period.

- Hitrost prenosa informacij med predpomnilnikom in CPE mora biti zato v našem primeru petkrat večja kot pri necevovodni CPE.

- Pri načrtovanju cevovodne CPE je treba paziti, da od neke enote (register, ALE, …) ne zahtevamo, da bi v določeni urini periodi morala delati dve različni operaciji.

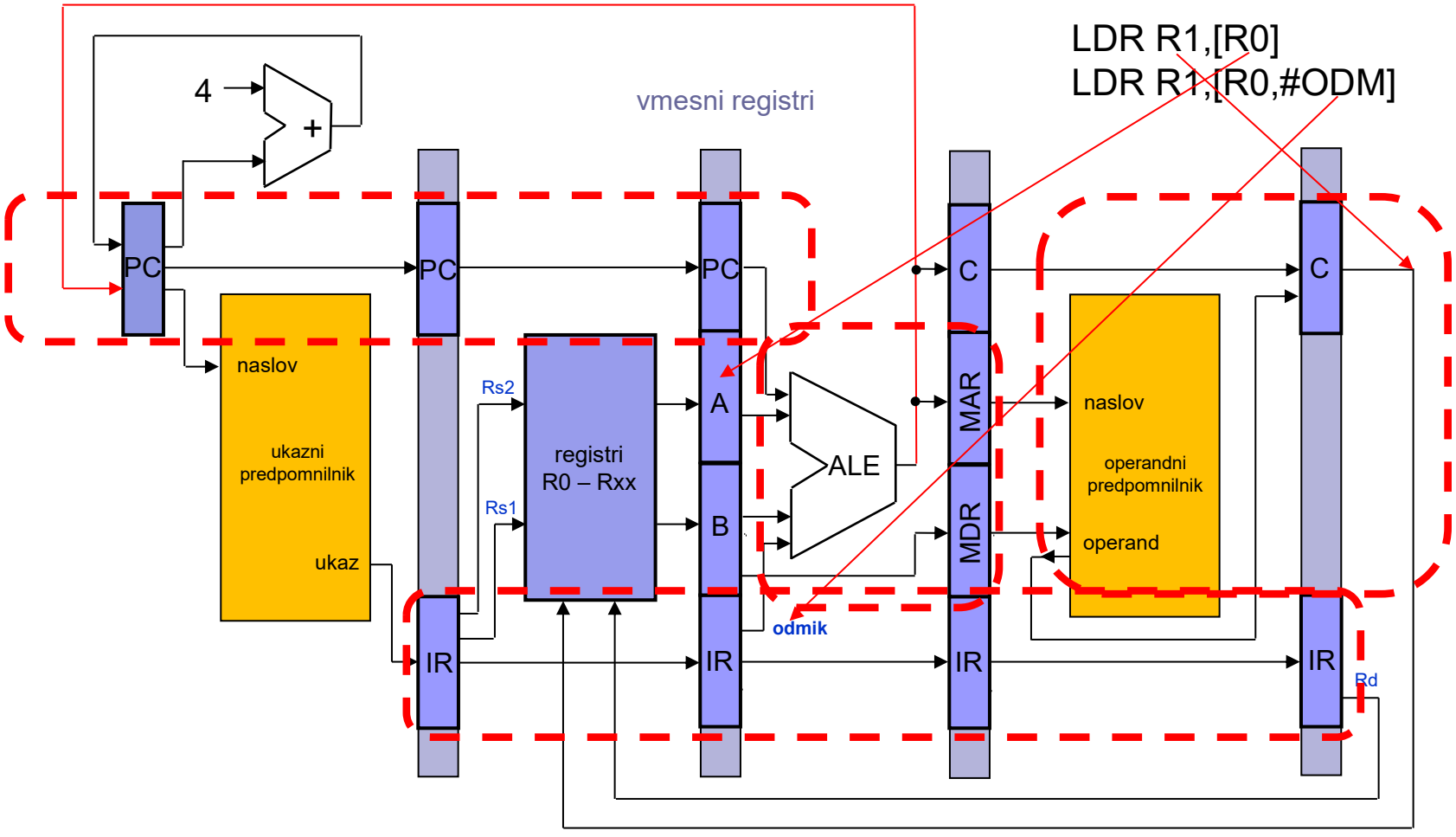
Primer zgradbe 5-stopenjske cevovodne CPE

(ALE ukaz: npr. ADD R1,R2,R3)



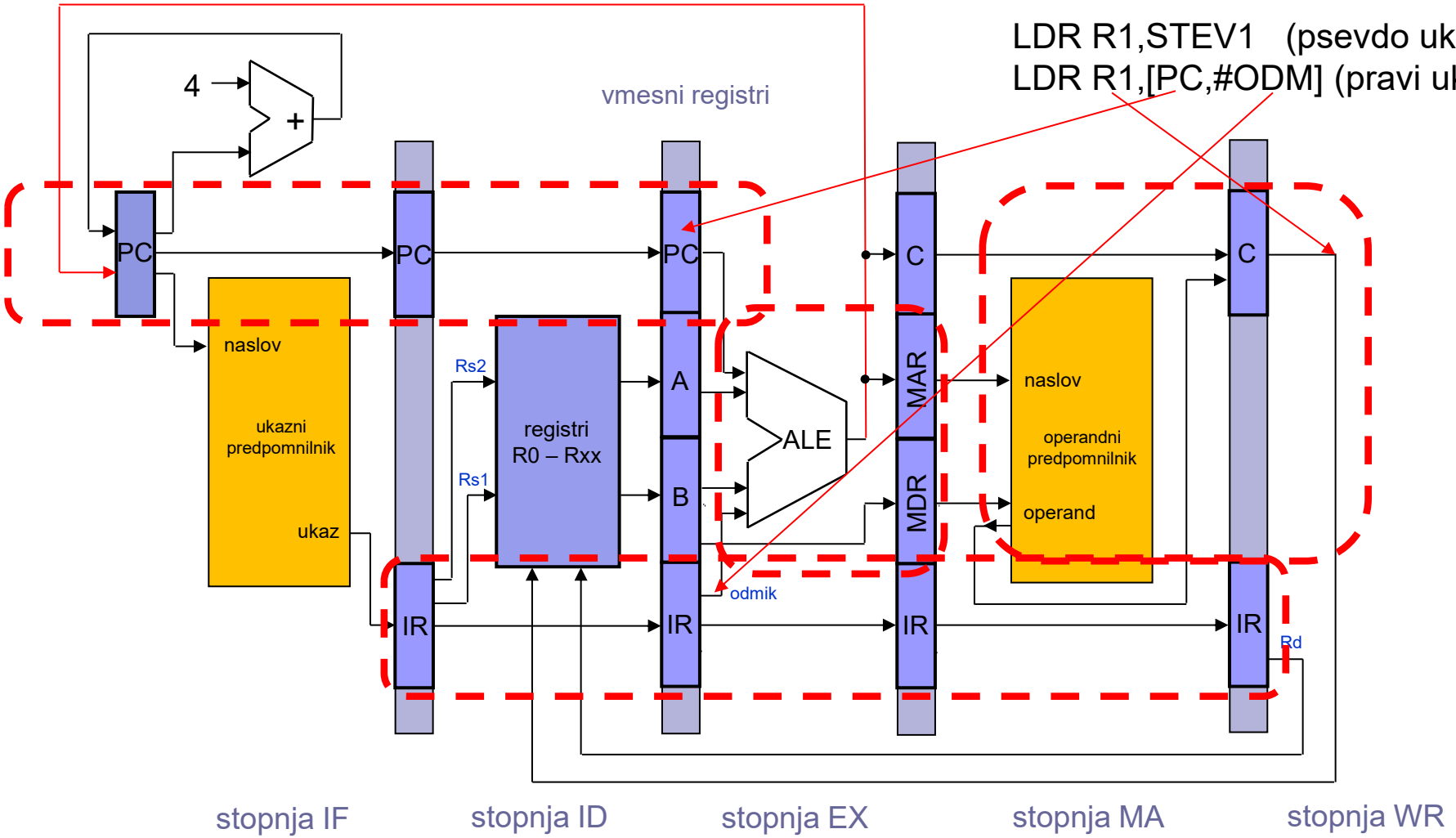
Primer zgradbe 5-stopenjske cevovodne CPE

(možnost izračuna skočnega naslova in naslova za LOAD/STORE v ALE v stopnji EX)



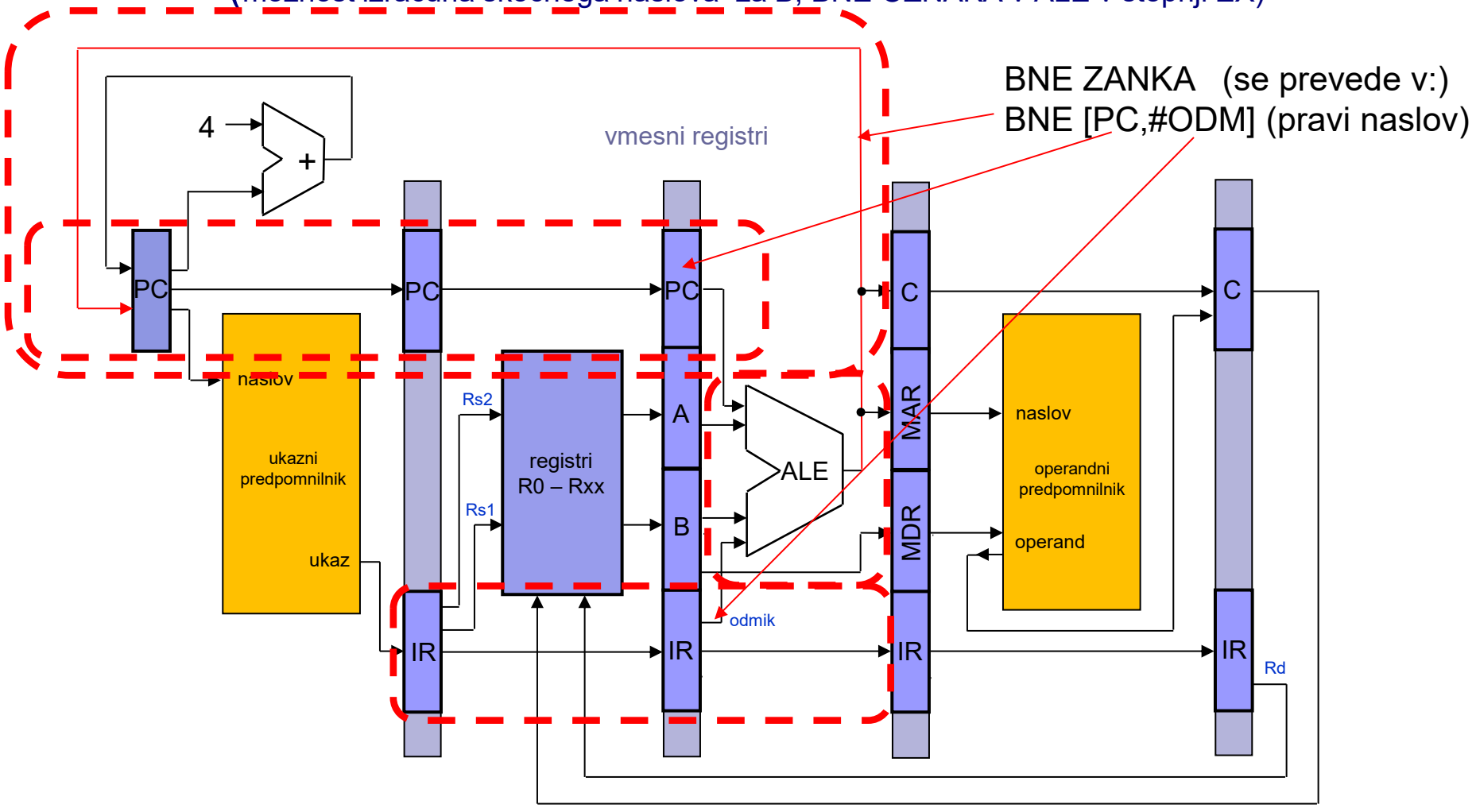
Primer zgradbe 5-stopenjske cevovodne CPE

(možnost izračuna skočnega naslova in naslova za LOAD/STORE v ALE v stopnji EX)



Primer zgradbe 5-stopenjske cevovodne CPE

(možnost izračuna skočnega naslova za B, BNE OZNAKA v ALE v stopnji EX)

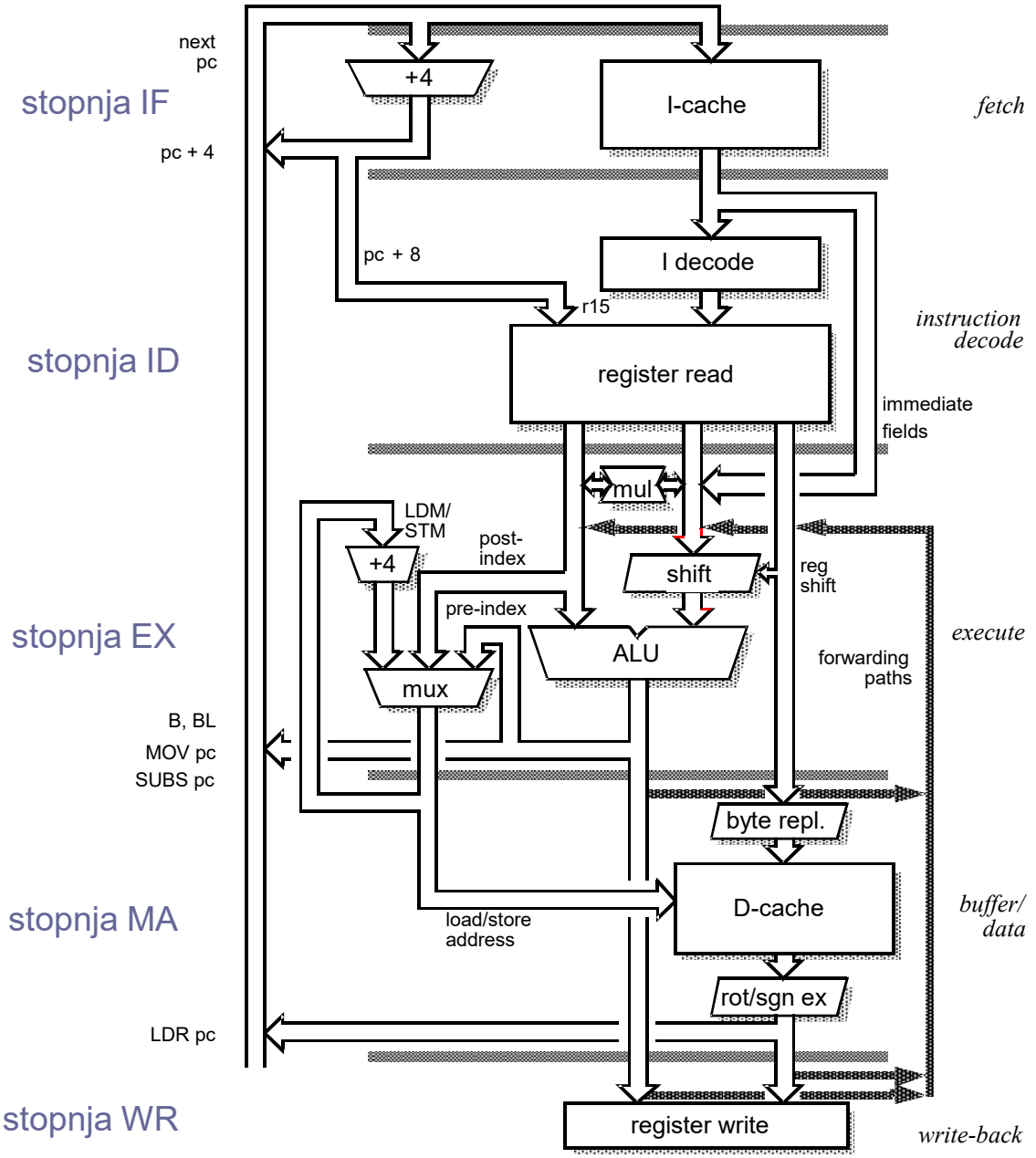


Primer zgradbe 5-stopenjske cevovodne CPE

- Cevovod ima 5 stopenj, med njimi so vmesni registri, v katere se shranijo rezultati podoperacij vsake stopnje, ki jih potrebujejo druge stopnje.
- V stopnji IF se prebere ukaz, ki se prenese v ukazni register in za 4 poveča vsebina programskega števca PC(ukazi so dolgi 4 bajte).
- Programski števec je potrebno povečati v stopnji IF zato, ker se vsako urino periodo iz ukaznega predpomnilnika prevzame nov ukaz.

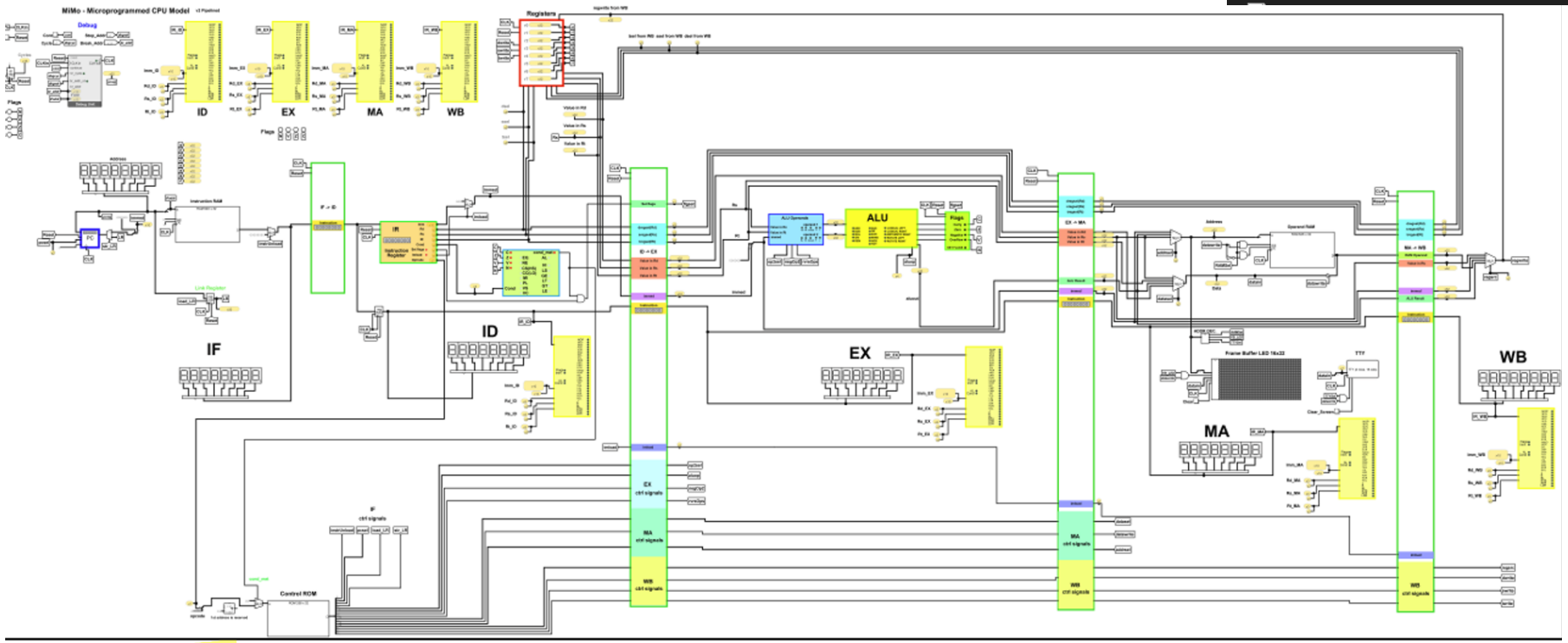
- Naslov ukaza, ki se izvaja (vsebina PC), se shranjuje v vmesne registre, ker je pri skočnih ukazih potreben v stopnji EX.
- Pri skočnih ukazih se v PC namreč vpiše nova vrednost (skočni naslov), ki se izračuna v ALE v stopnji EX.
- Naslov operanda pri ukazih LOAD/STORE (posredno naslavljanje) se prav tako izračuna v ALE v stopnji EX.
- Vsaka stopnja izvršuje drug ukaz, zato je treba v vmesne registre IR med vsemi stopnjami vedno shranjevati tudi ukaz, ki se v stopnji IF vsako urino periodo prebere iz ukaznega predpomnilnika.

Konkretni primer zgradbe 5-stopenjske cevovodne CPE
FRI SMS Atmel 9260 ARMv5



MiMo v2 - 5. st. cevovod v Logisimu

mimo_32bit_v2.1 - Zaklenitev.circ
mimo_32bit_v2.2 - Premoscanje.circ
mimo_32bit_v2.3 - Predikcije.circ
mimo_32bit_v2.circ




```
loop:       @ stall                                          | forwarding
mov r3, #3  @ 5, 22                                          | 5, 17
ldr r1, [r2]    @ 6, 23                                      | 6, 18
add r1, r1, #1  @ 10, 27                                     | 8, 20 (here
add r7, r7, #1  @ 11, 28                                     | 9, 21
str r2, r1  @ 14 (written to operand memory on cycle 13, but left pip
subs r4, r3, r1 @ 15, 32                                     | 11, 23
add r5, r5, #1  @ 17, 34                                     | 12, 24
add r7, r7, #1  @ 18, 35                                     | 13, 25
add r6, r1, r4  @ 19, 36                                     | 14, 26
jne loop    @ 20, 37                                         | 15, 27
```

Primerjava zaklepanja in premoščanja:
37 (zaklepanje) in 27 (premoščanje) urinih period za izvedbo programa

test1-nops_needed.txt
test2-zaklenitev_with_no_nops.txt
test3-operand_forwarding.txt
test4-jumps_in_op_forwarding.txt
test5-stall_vs_forwarding.txt

https://github.com/LAPSyLAB/MiMo_Student_Release/tree/main/MiMo_v2_Pipelined_versions

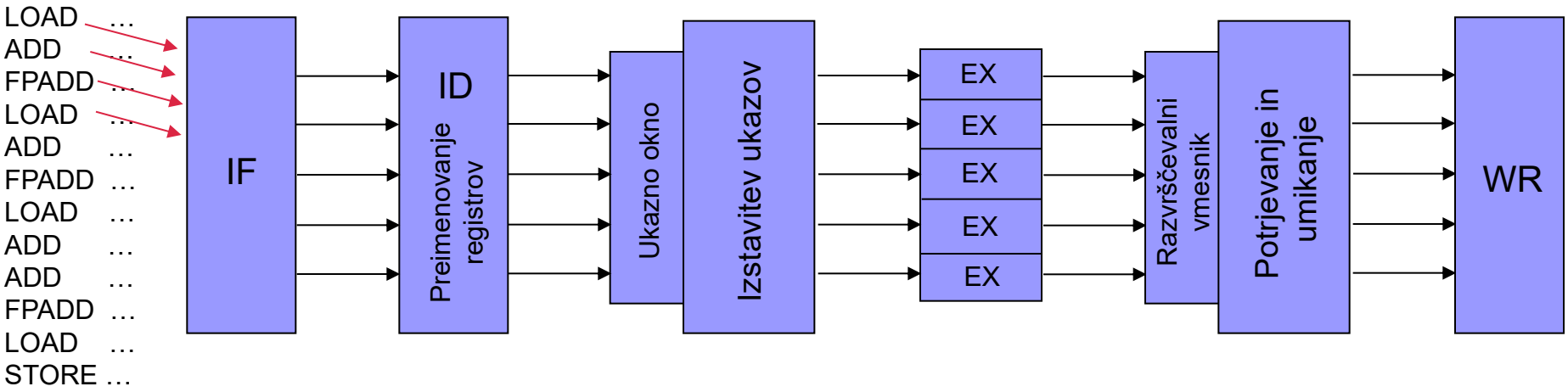
6.8 Večizstavitveni procesorji

- S cevovodno CPE in z odpravljanjem cevovodnih nevarnosti lahko dosežemo CPI, ki je blizu 1.
- Če želimo CPI zmanjšati pod 1, moramo v vsaki urini periodi prevzeti več ukazov (in jih tudi izvesti).
- Take procesorje označujemo z izrazom večizstavitveni procesorji in jih delimo na dve vrsti:
  - Superskalarni procesorji – ukaze, ki se paralelno izvajajo, določa logika v procesorju
  - VLIW procesorji – ukaze, ki se paralelno izvajajo, določa program (prevajalnik)

Superskalarni procesorji

Superskalarni procesor je cevovodni procesor, ki je sposoben hkrati prevzemati, dekodirati in izvrševati več ukazov.

- Število prevzetih in izstavljenih ukazov v eni urini periodi se med izvajanjem programa dinamično spreminja in ga določa logika v procesorju.
- Procesor, ki lahko izstavi največ n ukazov se imenuje *n-kratni* superskalarni procesor.
- Istočasno superskalarno delovanje zahteva dodatne vmesnike in dodatne stopnje za ugotavljanje medsebojnih odvisnosti, potrjevanje in morebitno umikanje rezultatov ->

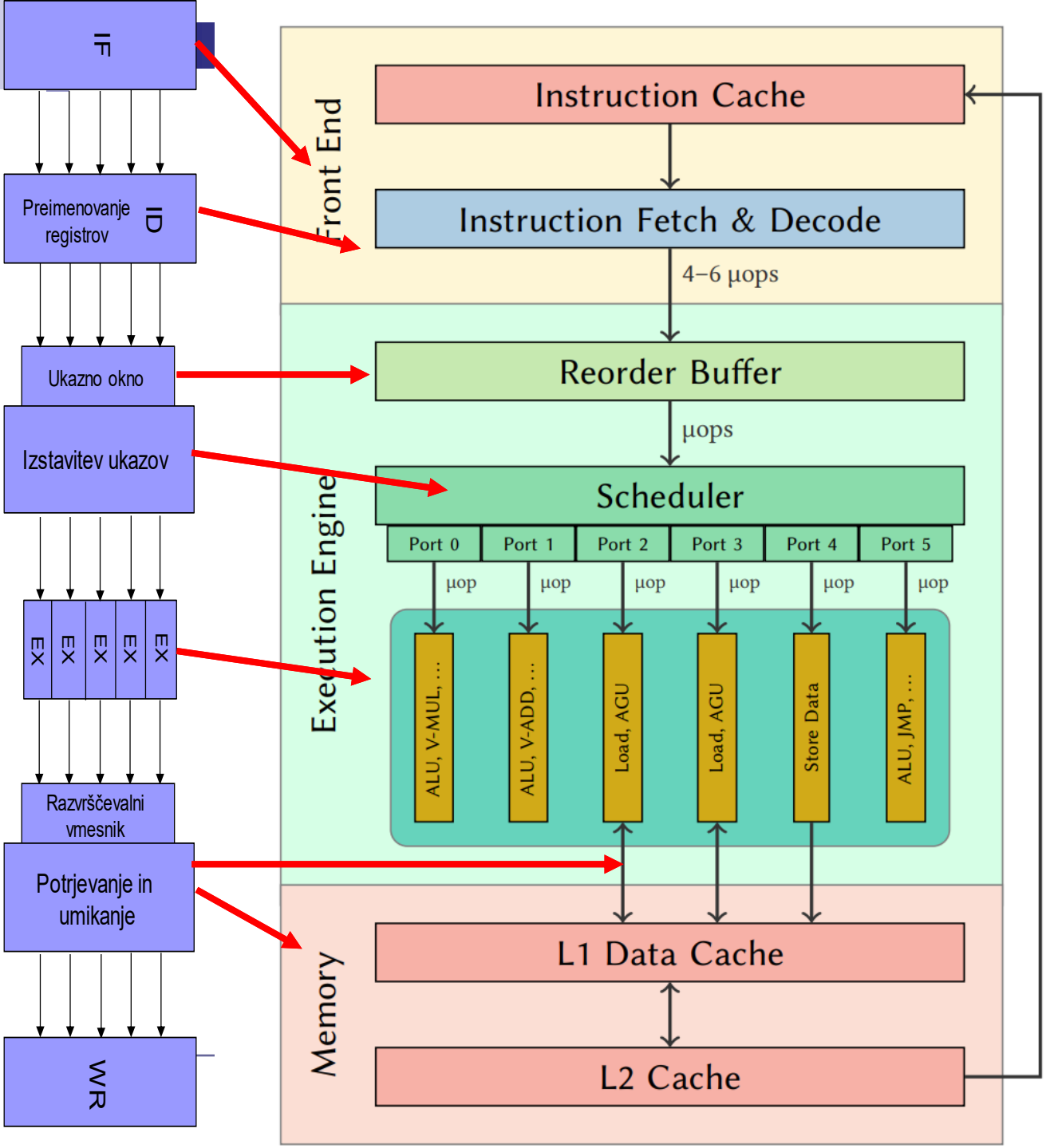


Poenostavljena shema superskalarnega procesorja, ki ima za osnovo petstopenjski cevovod

Primeri EX funkcijskih enot :

- Enote za operacije v fiksni/plavajoči vejici
- Ena od funkcijskih enot v stopnji EX je stopnja MA (funkcijska enota LOAD/STORE ali ločeni funkcijski enoti LOAD in STORE).

Poenostavljen primer superskalarnega procesorja: Intel Core i7



1.Instruction Fetch (16bajtov)

2.Predecode Stage (bajti->x86 ukaze)

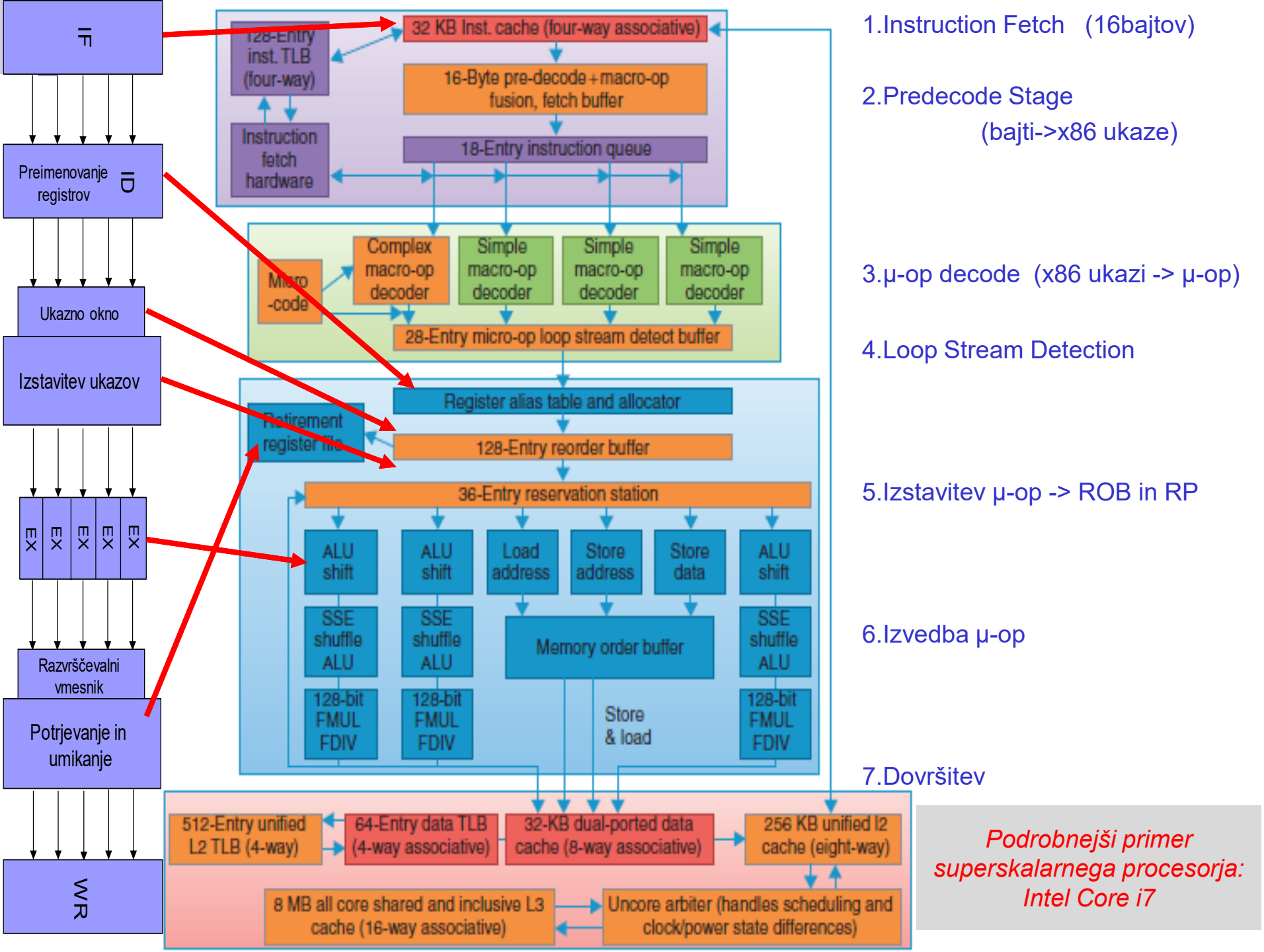
3.µ-op decode (x86 ukazi -> µ-op)

4.Loop Stream Detection

5.Izstavitev µ-op -> ROB in RP

6.Izvedba µ-op

7.Dovršitev



1.Instruction Fetch (16bajtov)

2.Predecode Stage
(bajti->x86 ukaze)

3.μ-op decode (x86 ukazi -> μ-op)

4.Loop Stream Detection

5.Izstavitev μ-op -> ROB in RP

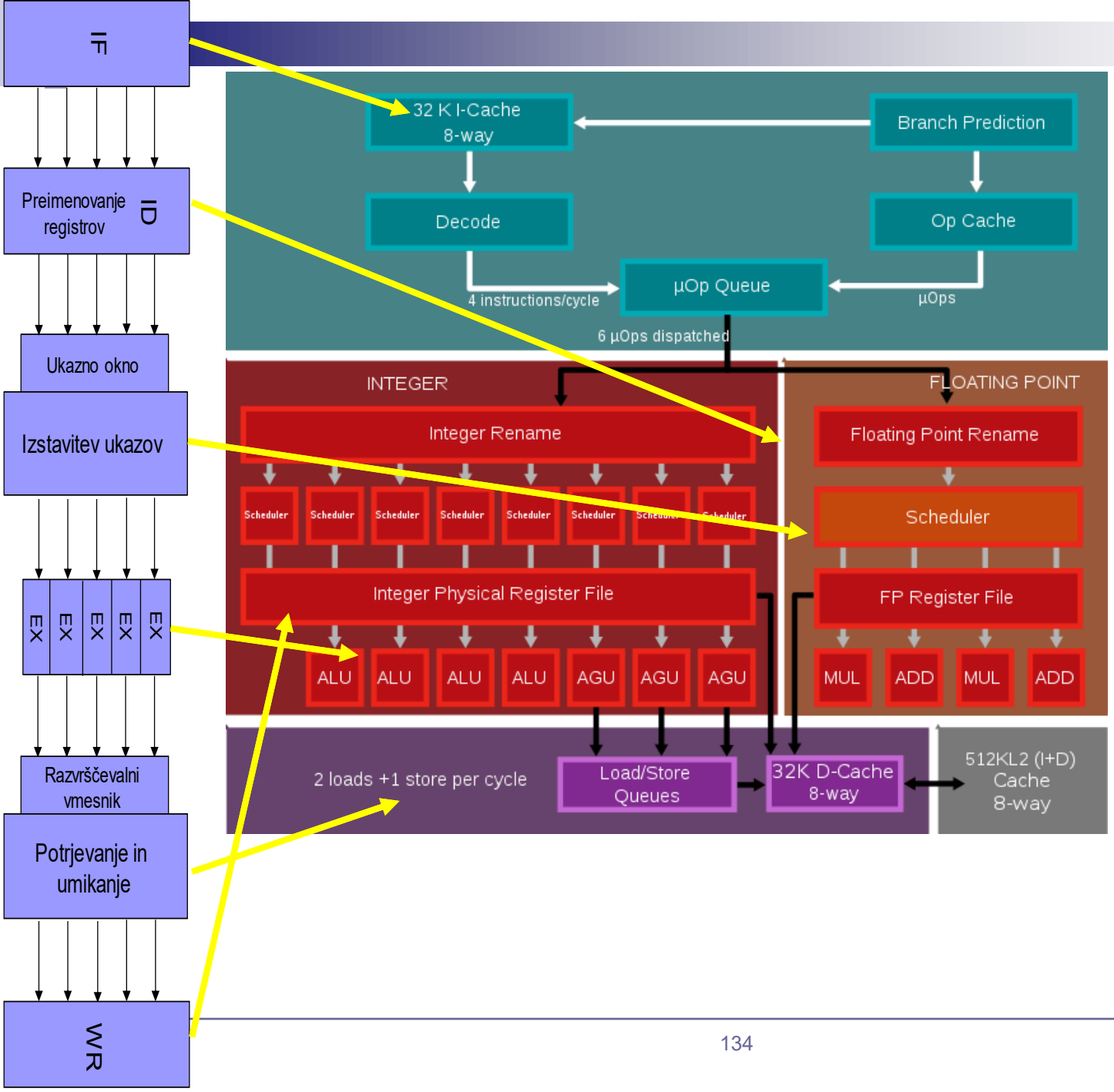
6.Izvedba μ-op

7.Dovršitev

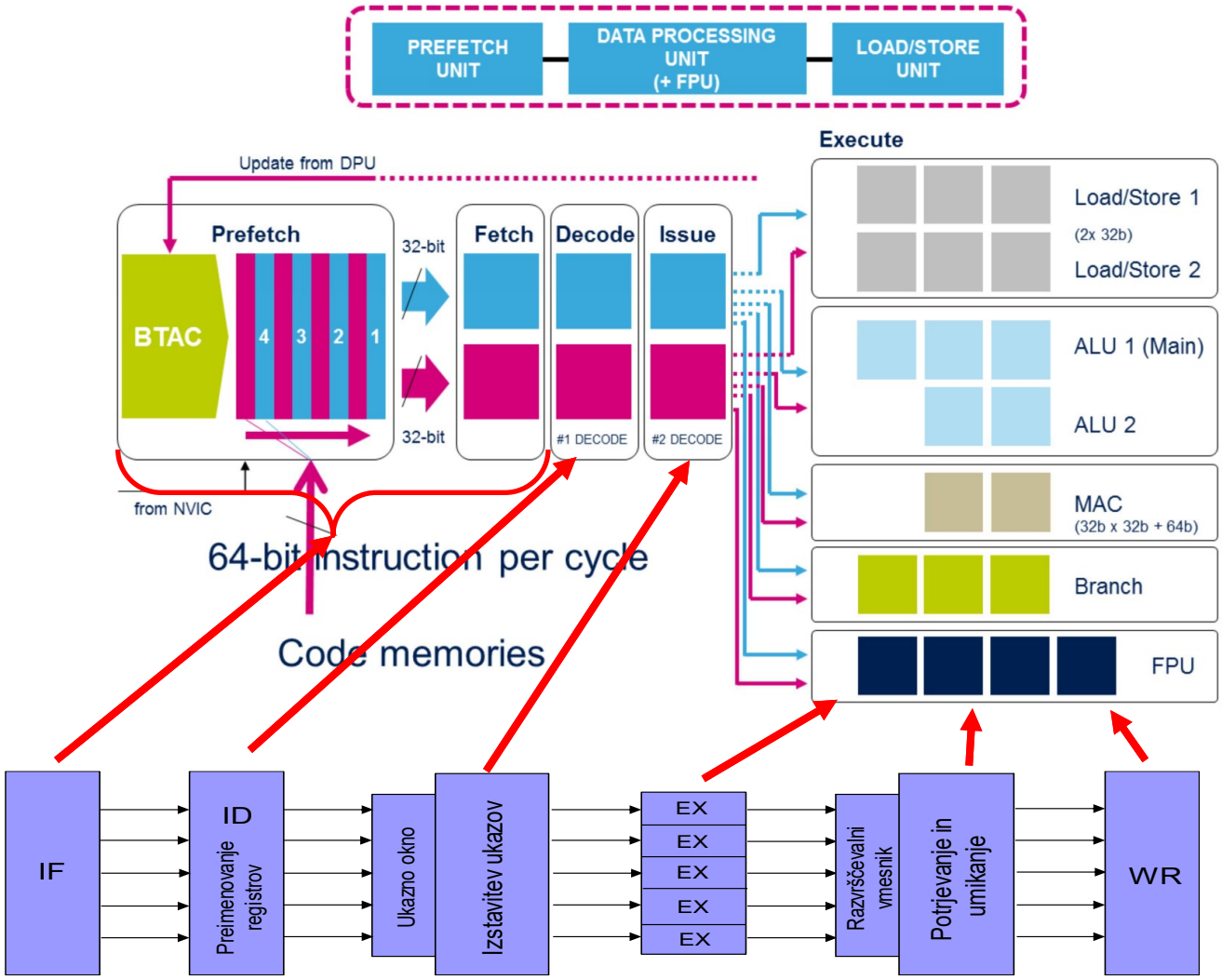
Podrobnejši primer superskalarnega procesorja: Intel Core i7

Primer superskalarnega procesorja:

AMD Zen 2



ARM Cortex-M7 → Dual-issue



ARM Cortex M7

Primer dvoizstavitvenega enostavnejšega procesorja

STM32H750 – Arm Cortex M7 – dvo-izstavitveni procesor

VLIW procesor

VLIW (Very Long Instruction Word) procesorji izvršujejo dolge ukaze, ki so sestavljeni iz več običajnih strojnih ukazov, ki jih procesor lahko paralelno izvršuje v različnih funkcijskih enotah.

- V dolgem ukazu izvršuje vsaka funkcijska enota svoj ukaz.

VLIW ukaz sestavljajo ukazi za posamezne funkcijske enote

Primer konkretne sestave VLIW ukaza

- Prevajalnik v programu išče med seboj neodvisne ukaze za funkcijske enote in z njimi sestavlja dolge ukaze.
- Število ukazov, ki so prevzeti in izstavljeni v eni urini periodi je določeno s programom in se med delovanjem ne spreminja.
- Če prevajalnik ne najde dovolj ukazov za vse funkcijske enote, da enoti, za katero ni našel ukaza, ukaz NOP (No OPeration).

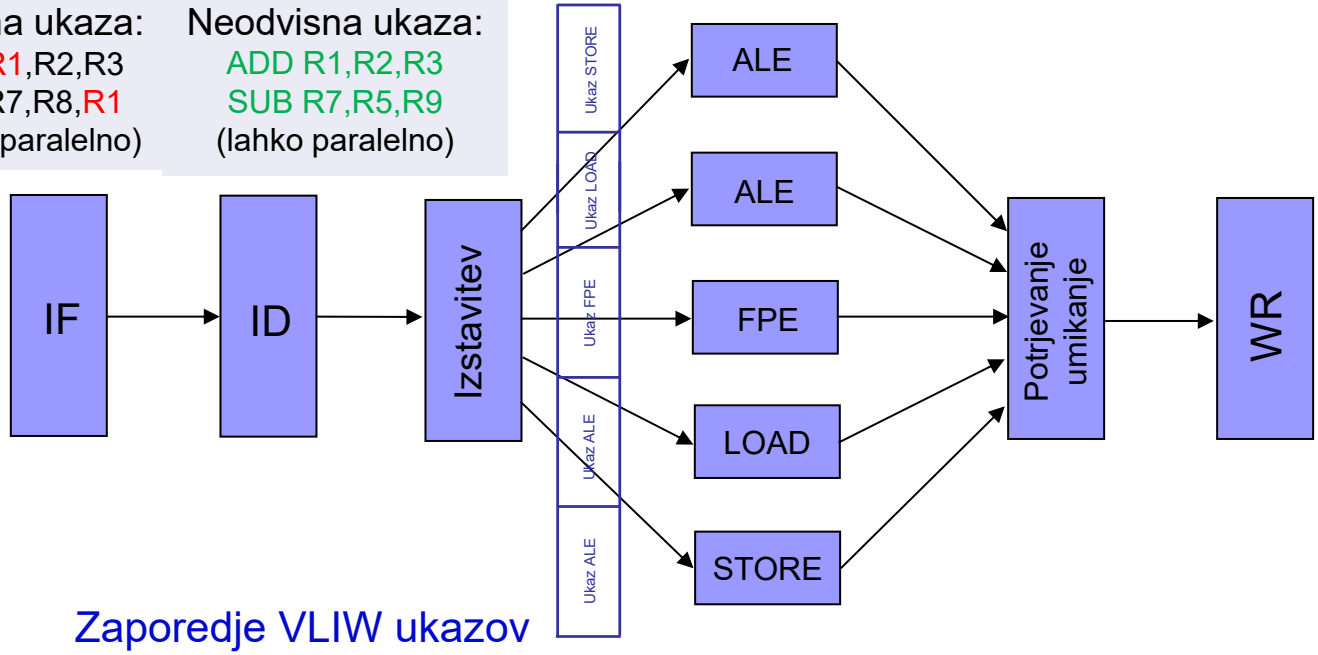
Prevajalnik v programu išče med seboj neodvisne ukaze za funkcijske enote in z njimi sestavlja dolge ukaze.
Če ne najde ukaze za posamezno funkcijsko enoto, vstavi ukaz NOP („-“ v VLIW ukazu spodaj).

Program

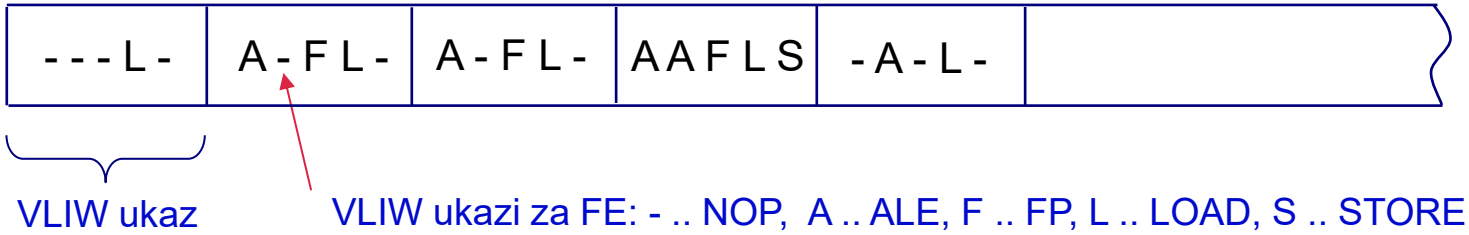
```
LOAD  …
ADD   …
FPADD …
LOAD  …
ADD   …
FPADD …
LOAD  …
ADD   …
ADD   …
FPADD …
LOAD  …
STORE …
```

Odvisna ukaza:
ADD R1,R2,R3
SUB R7,R8,R1
(ne gre paralelno)

Neodvisna ukaza:
ADD R1,R2,R3
SUB R7,R5,R9
(lahko paralelno)

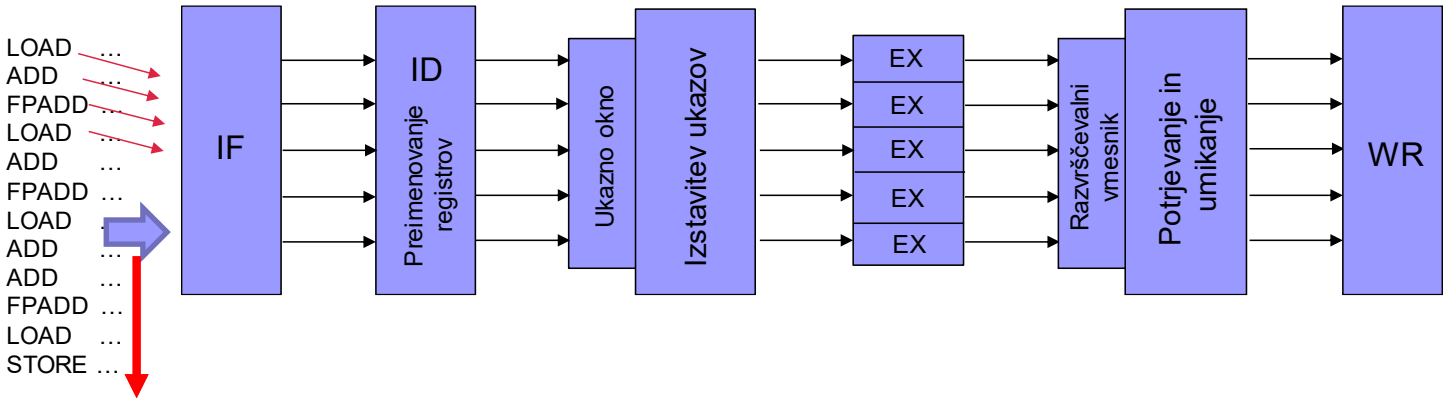


Zaporedje VLIW ukazov



Superskalarni procesor

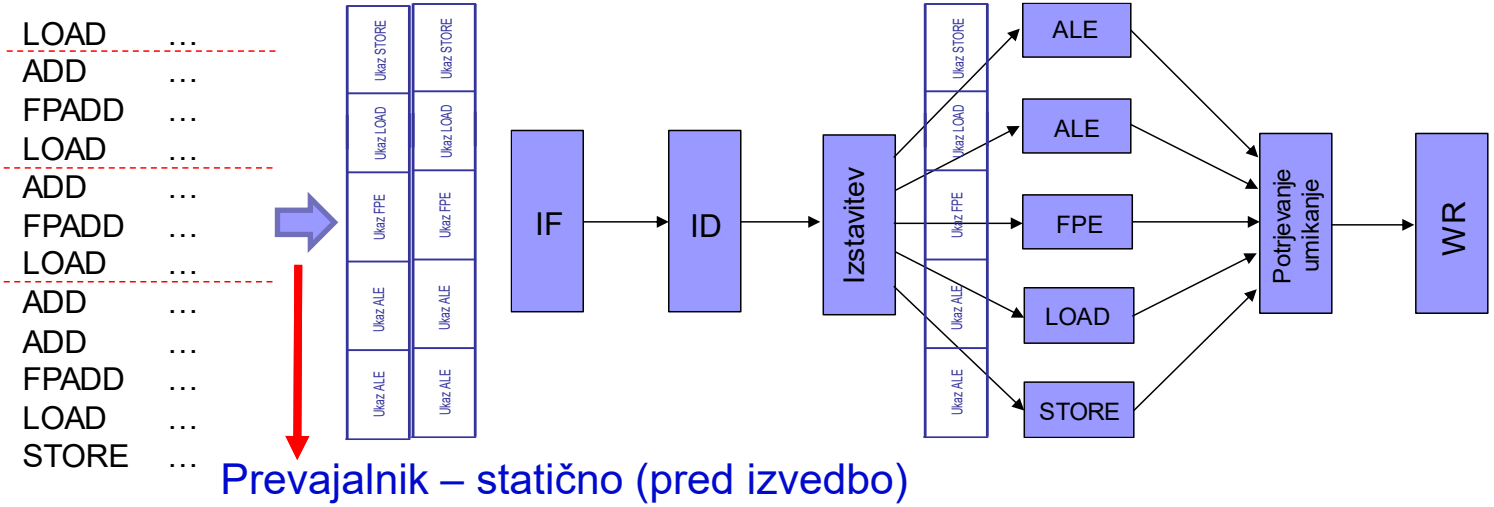
več ukazov hkrati



CPE – dinamično (med izvedbo)

VLIW procesor

dolgi ukaz (več krajših ukazov hkrati)



Prevajalnik – statično (pred izvedbo)

