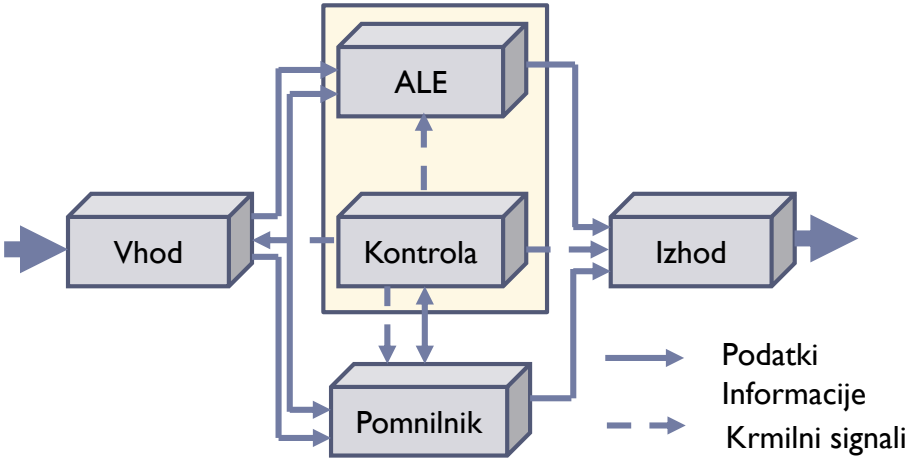
Digitalna vezja, BVS-RI

Mira TREBAR

P12 Načrtovanje procesorja

Uvod

- Digitalni računalnik (ang. digital computer) – strojna oprema, ki
  - izvaja operacije,
  - upravlja s podatki (binarna oblika),
  - sprejema odločitve.
- Enote računalnika (slika)
- Program – niz ukazov, ki se izvajajo.

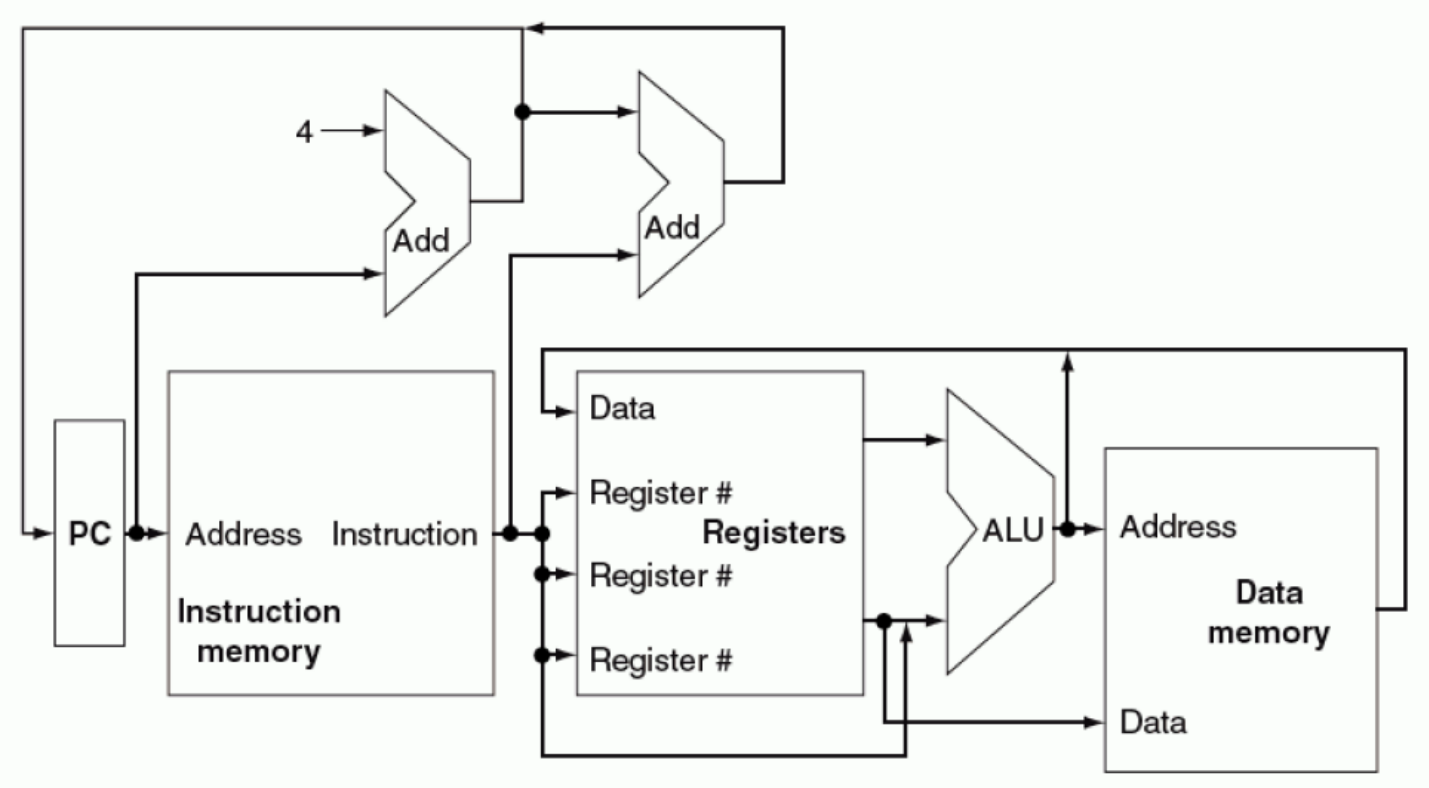


- Mikroprocesor (ang. microprocessor) – CPE je implementirana v enem integriranem vezju, čipu (v 70tih).
- Mikroračunalnik (ang. microcomputer) – mikroprocesor je združen s pomnilnikom in vhodno/izhodnimi vezji.
- Mikrokrmilnik (ang. microcontroller) – mikroračunalnik se s posebno strojno opremo nahaja v enem integriranem vezju in je namenjen krmiljenju določenih naprav.

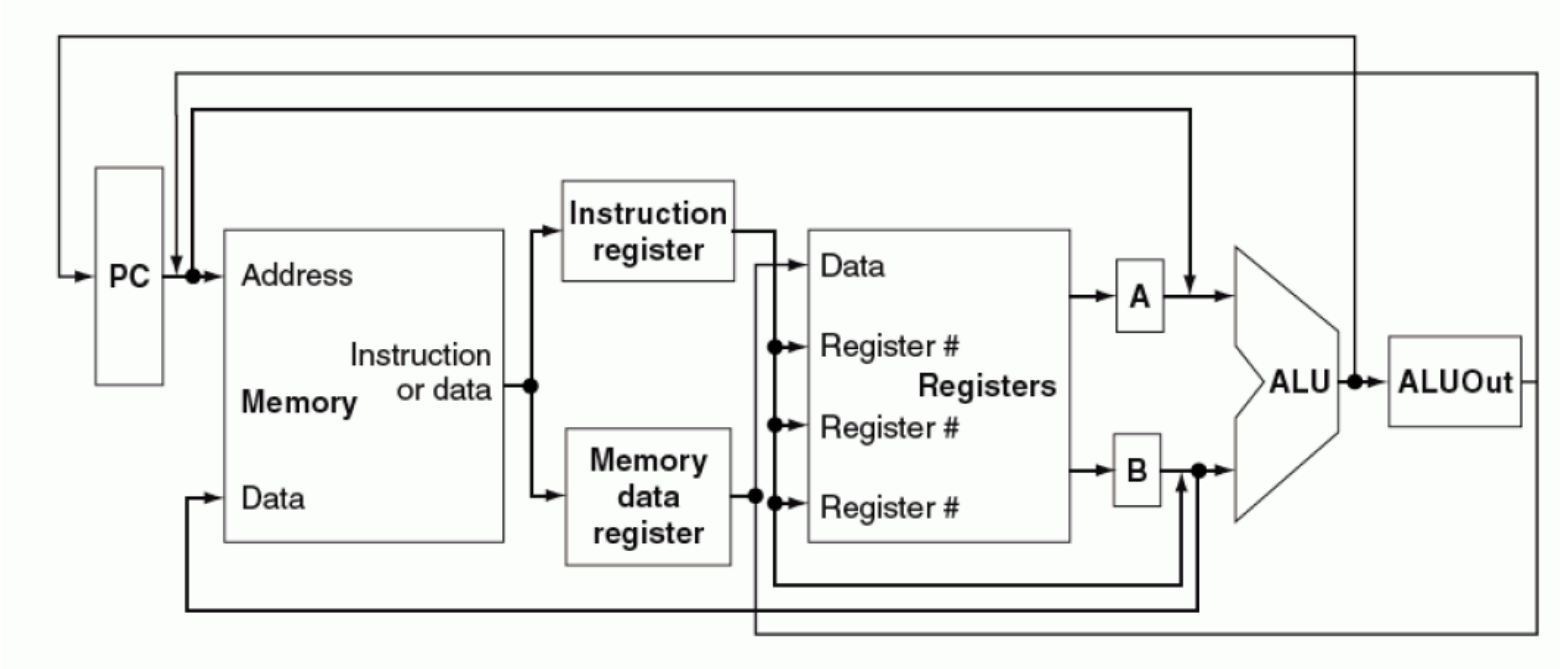
MIPS

- MIPS – 'millions of instructions per second'
- Arhitektura RISC (Reduced Instruction Set Computer)
- Zgodovina:
  - 1981 (John L.Hennessy, Stanford University) – začetki
  - 1984 (MIPS Computer Systems)– komercialna uporaba
  - R2000 je prvi 32-bitni procesor MIPS (1985-R3000, 1988-R4000)
  - 1999 – MIPS32 (32-bitni) in MIPS64 (64-bitni)
  - Leto 2002 – izdelanih >100M mikroprocesorjev, uporabljenih v proizvodih ATI Technologies, Cisco, NEC, Sony, Toshiba, ...
  - MIPS64 CPUs, MIPS I7200 Multhithread multicore processor
  - Microchip: PIC32 mikrokrmilniki so zasnovani na MIPS arhitekturi
- Uporaba:
  - Visoko-nivojski programski jeziki – (compiler) – Zbirni jezik – (Assembler) – binarni program v strojnem jeziku.
  - Koncept shranjenega programa – nabor ukazov
  - Procesor MIPS vsebuje tipičen nabor ukazov, ki je bil razvit po letu 1980.
- Vir: Procesor MIPS (Patterson&Hennessy, Computer Organization and Design, 3rd, ed, 2005

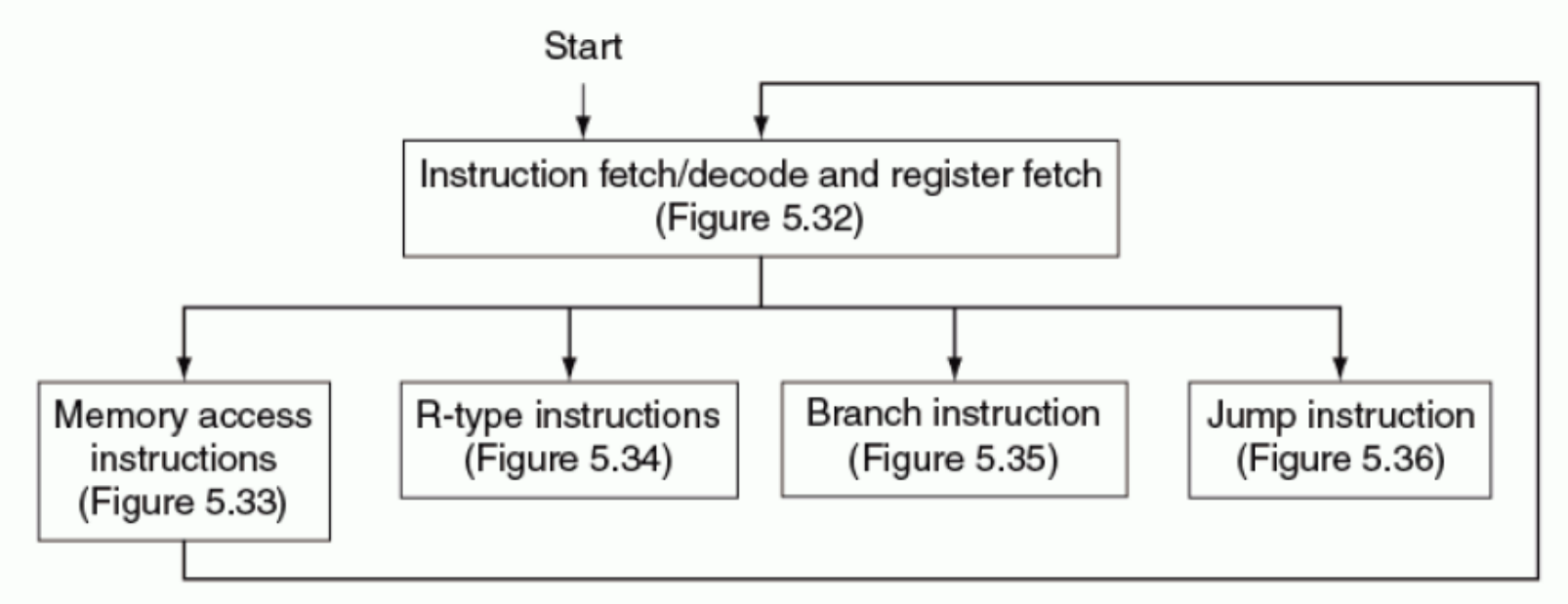
- MIPS (enocikelna izvedba) - ukazni in podatkovni pomnilnik sta ločena.
  - PC (naslov ukaza) - Dostava ukaza – Izvajanje ukaza



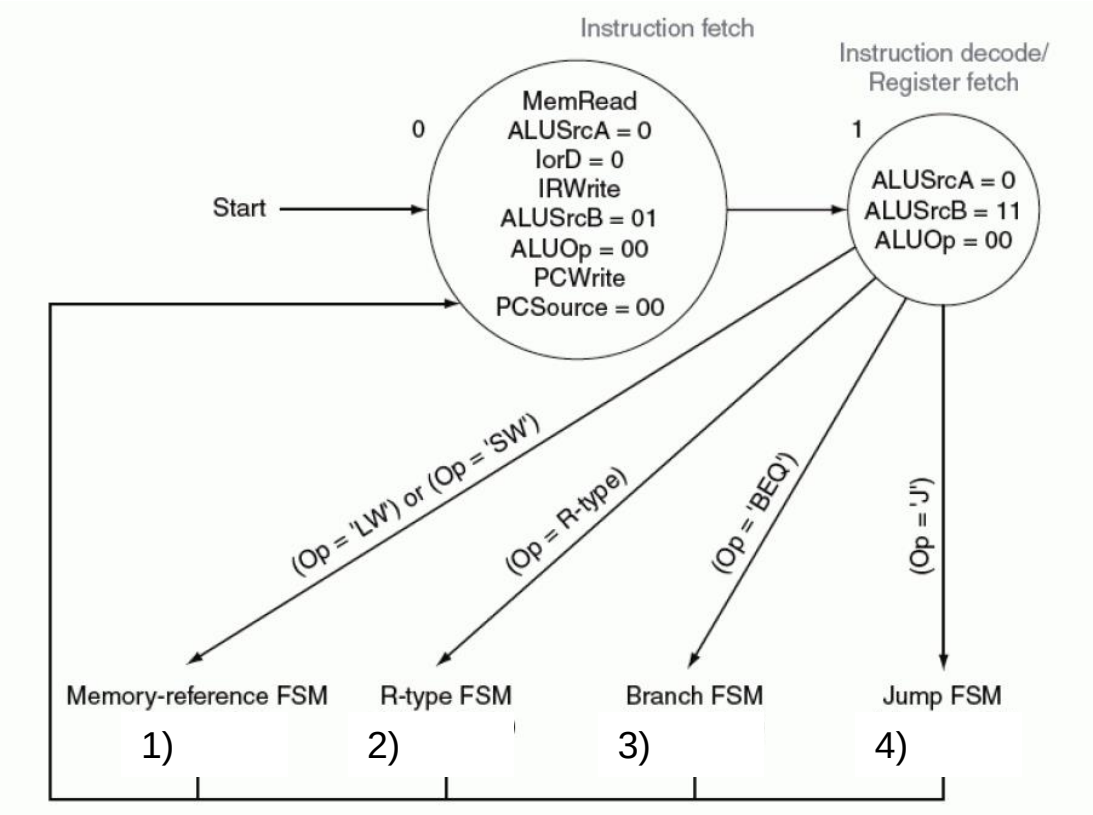
- MIPS (večcikelna izvedba) - skupen ukazni in podatkovni pomnilnik
  - PC (naslov ukaza) - Dostava ukaza, dekodiranje ukaza, registrska dostava
  - Izvajanje ukaza

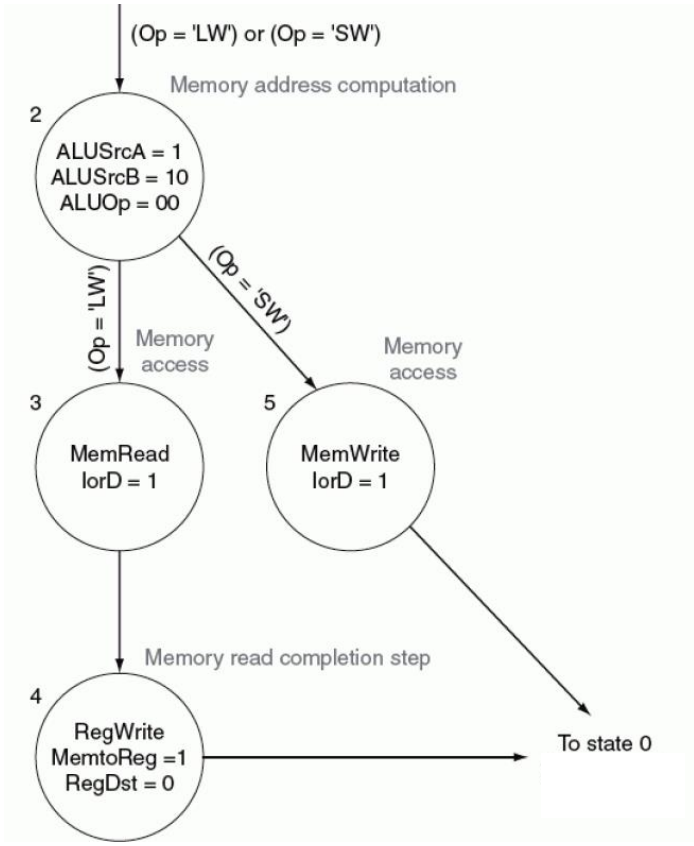


- Diagram prehajanja stanj (predstavitev avtomata)

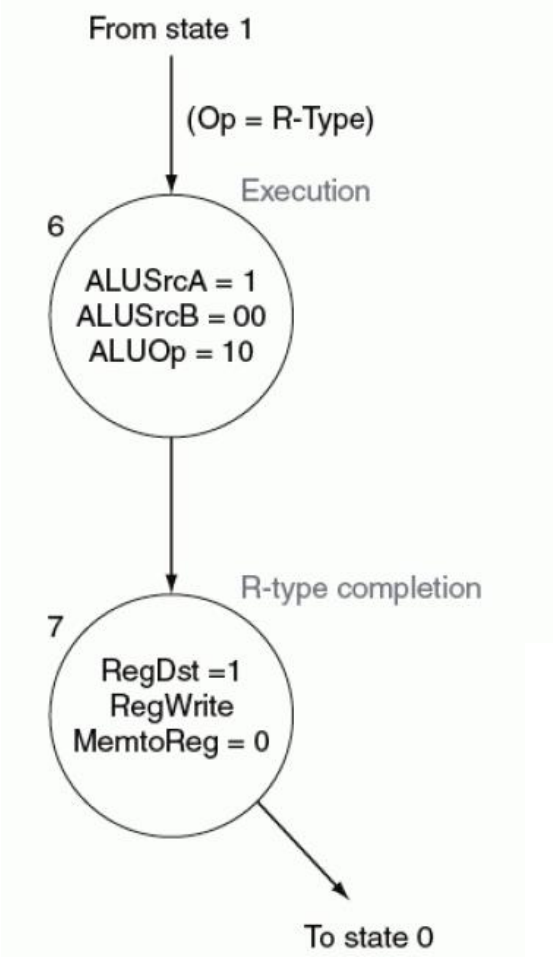


- Dostava ukaza, dekodiranje ukaza, registrska dostava, 1) ukaz LW ali SW

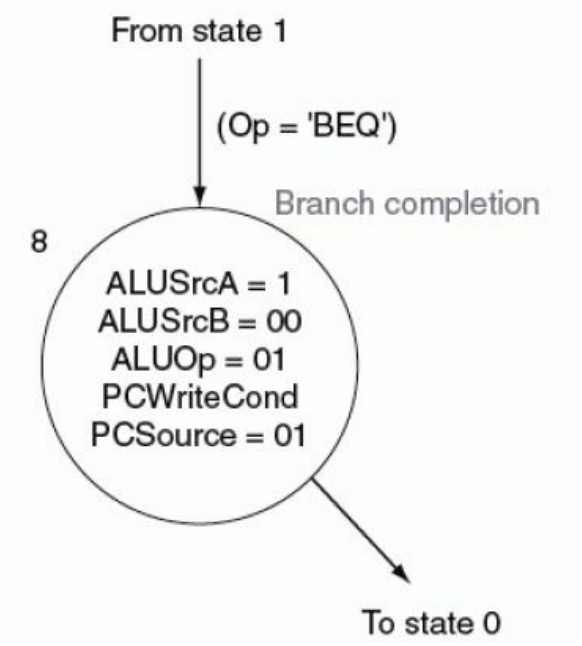




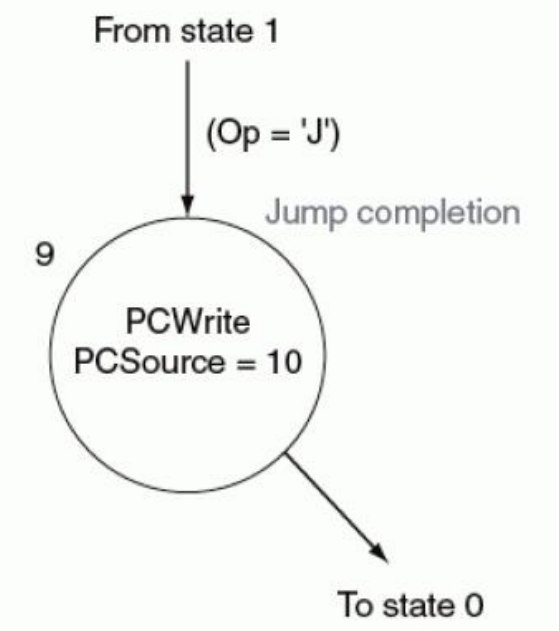
2) Ukaz tipa R



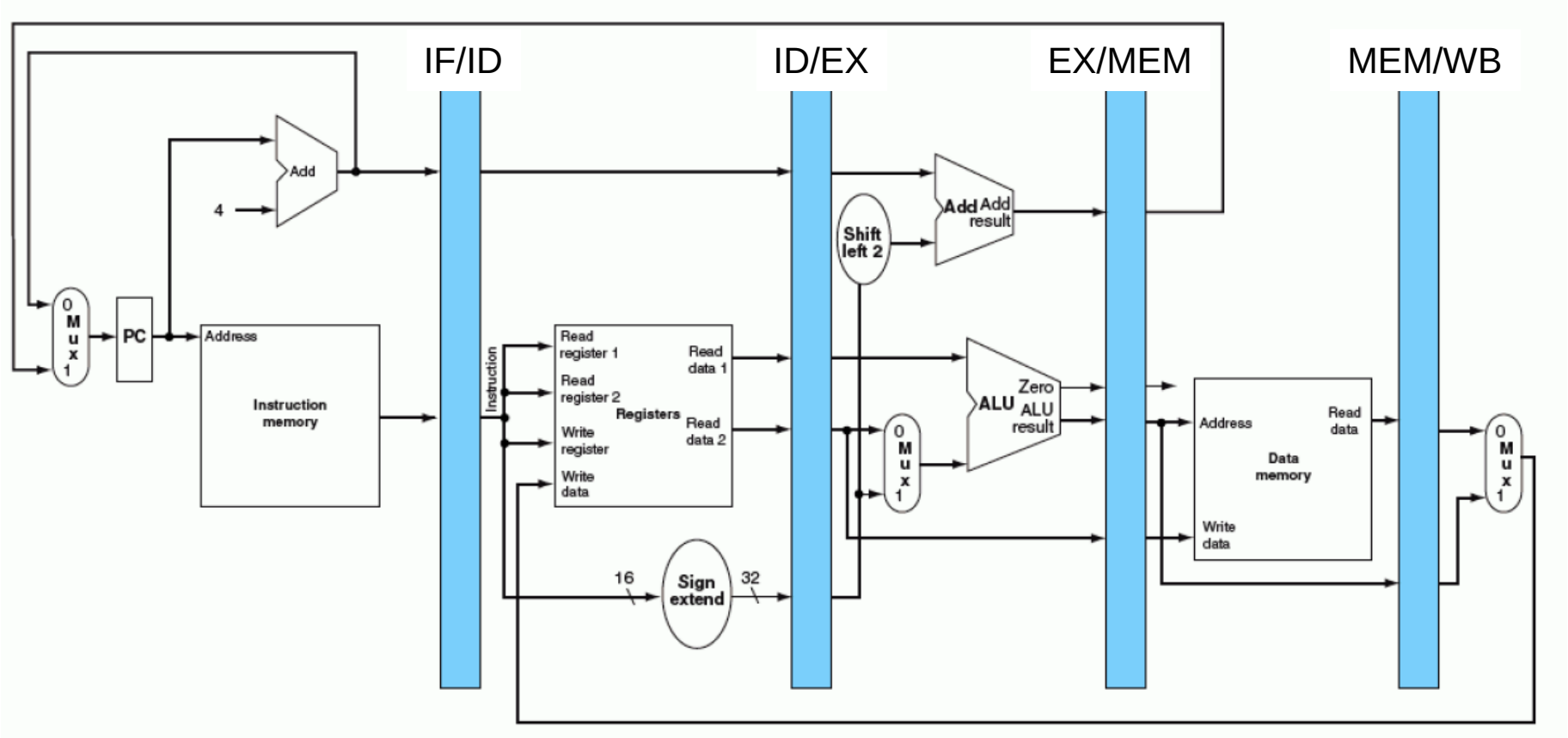
3) ukaz BEQ



4) Ukaz J

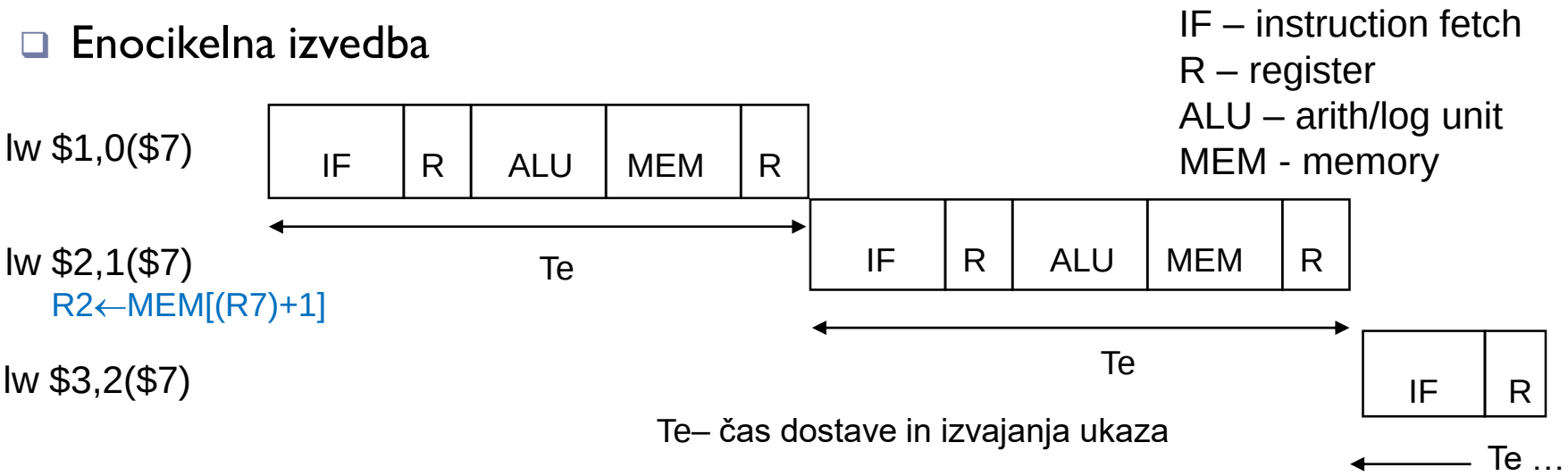


- MIPS (cevovodna izvedba) - ukazni in podatkovni pomnilnik sta ločena
  - Registri cevovoda ločujejo stopnje: IF/ID, ID/EX, EX/MEM, MEM/WB

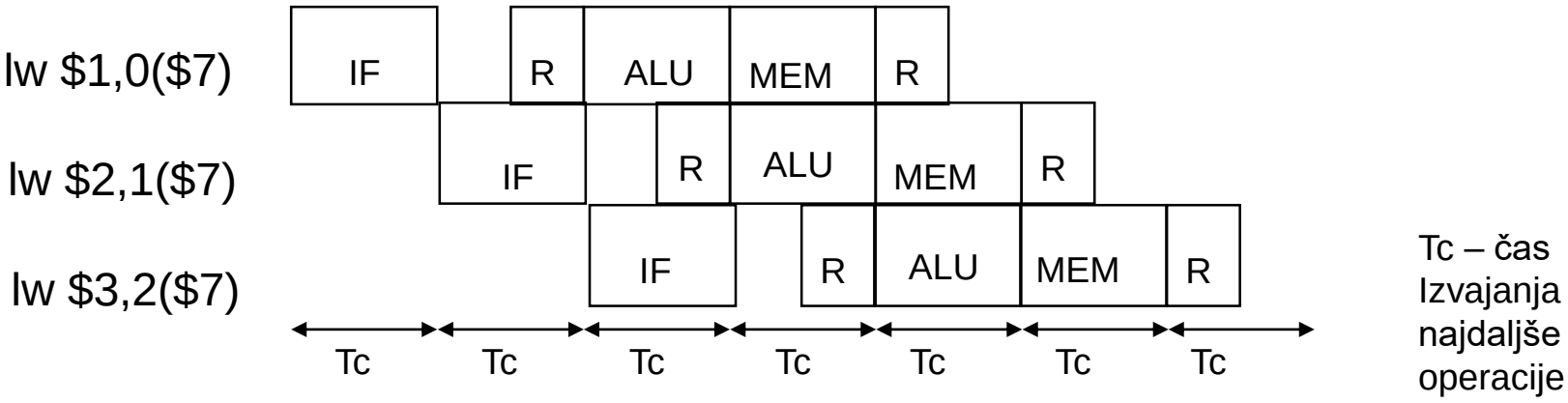


Primer nalaganja besede iz pomnilnika (load word)

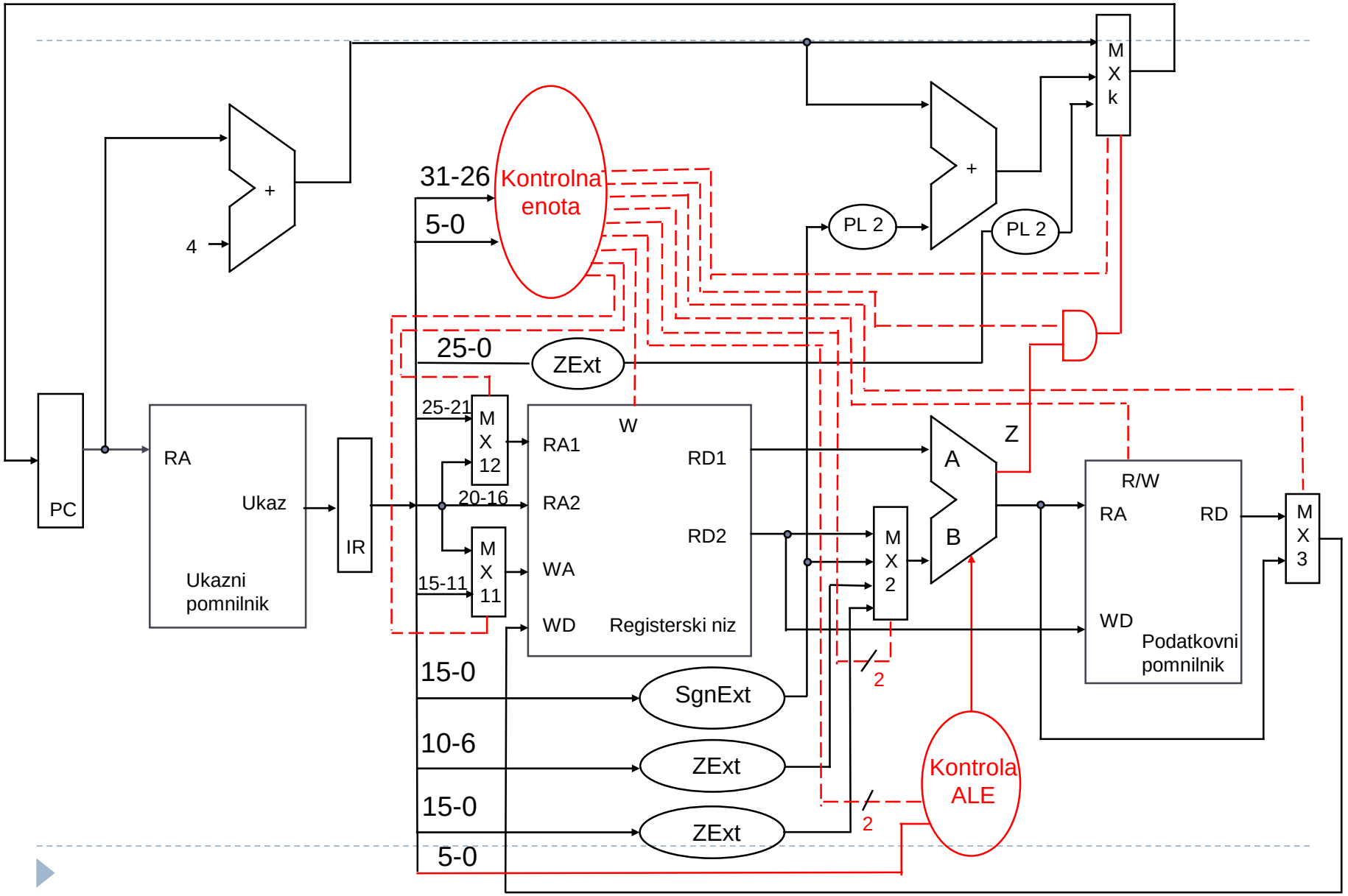
- Enocikelna izvedba



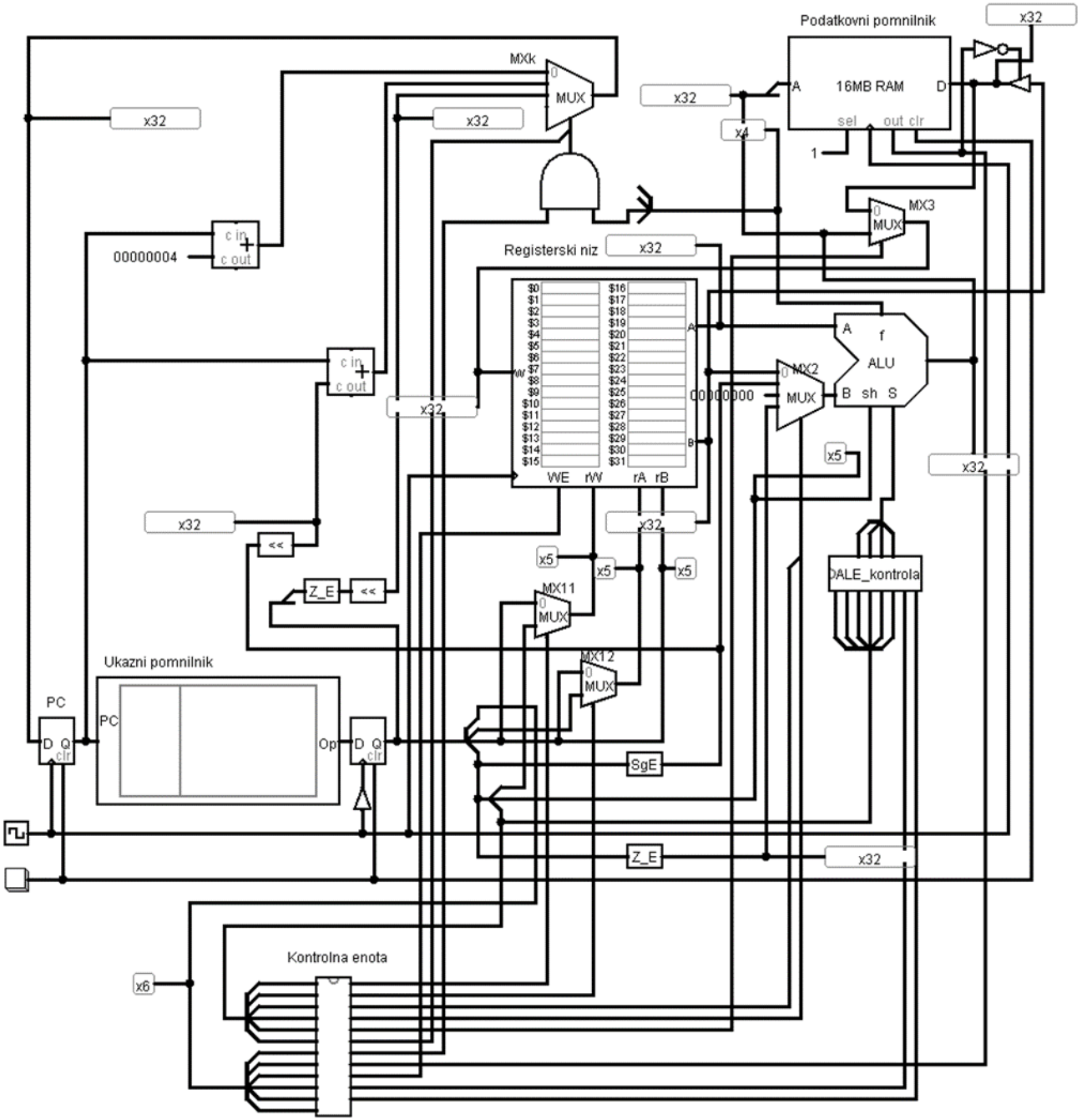
- Cevovodna izvedba



Procesor MIPS (podatkovne poti in krmilni signali)



Primer za 32-bitni MIPS (logisim)

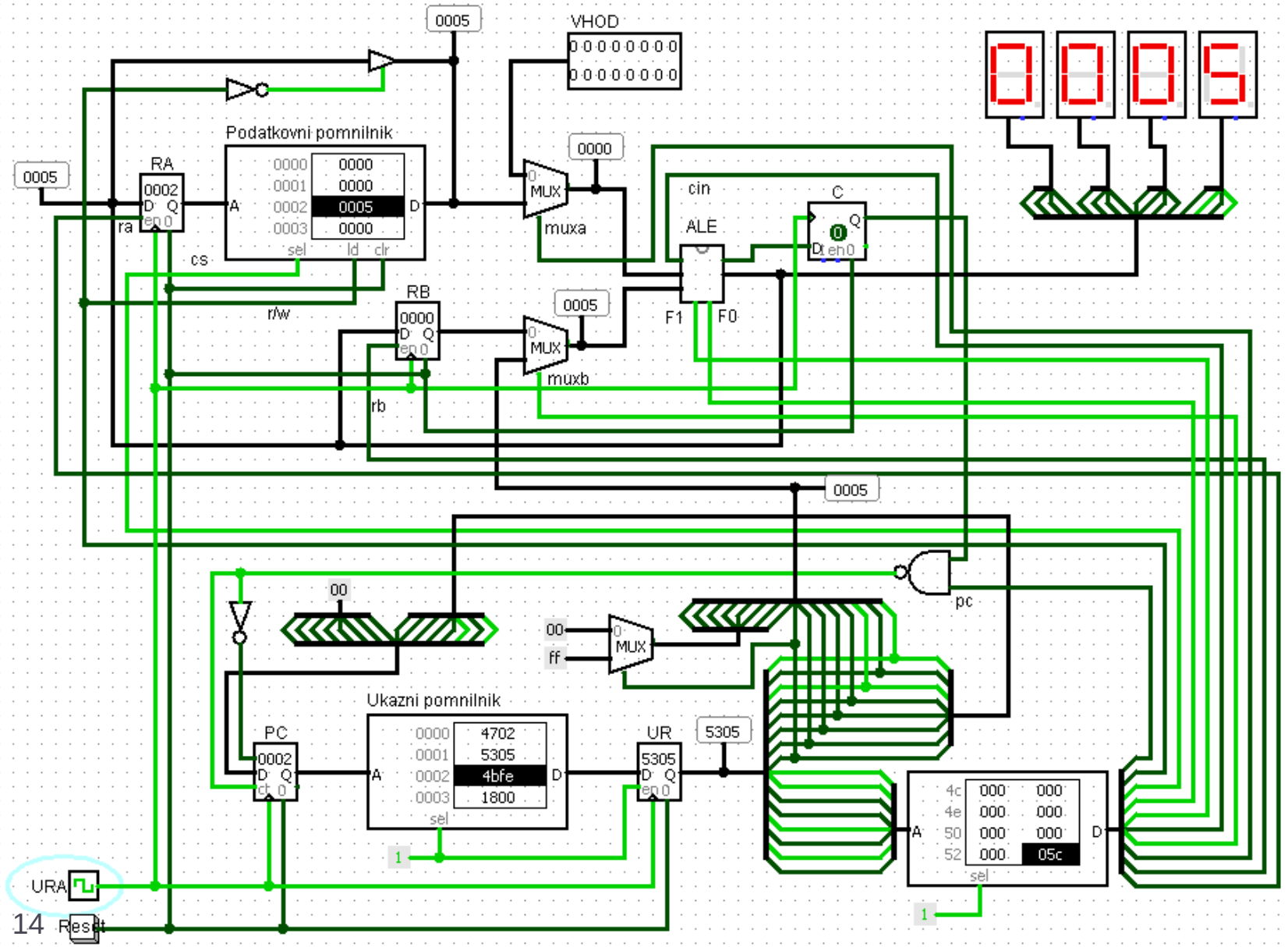


Primer: Načrtovanje in izvedba procesorja

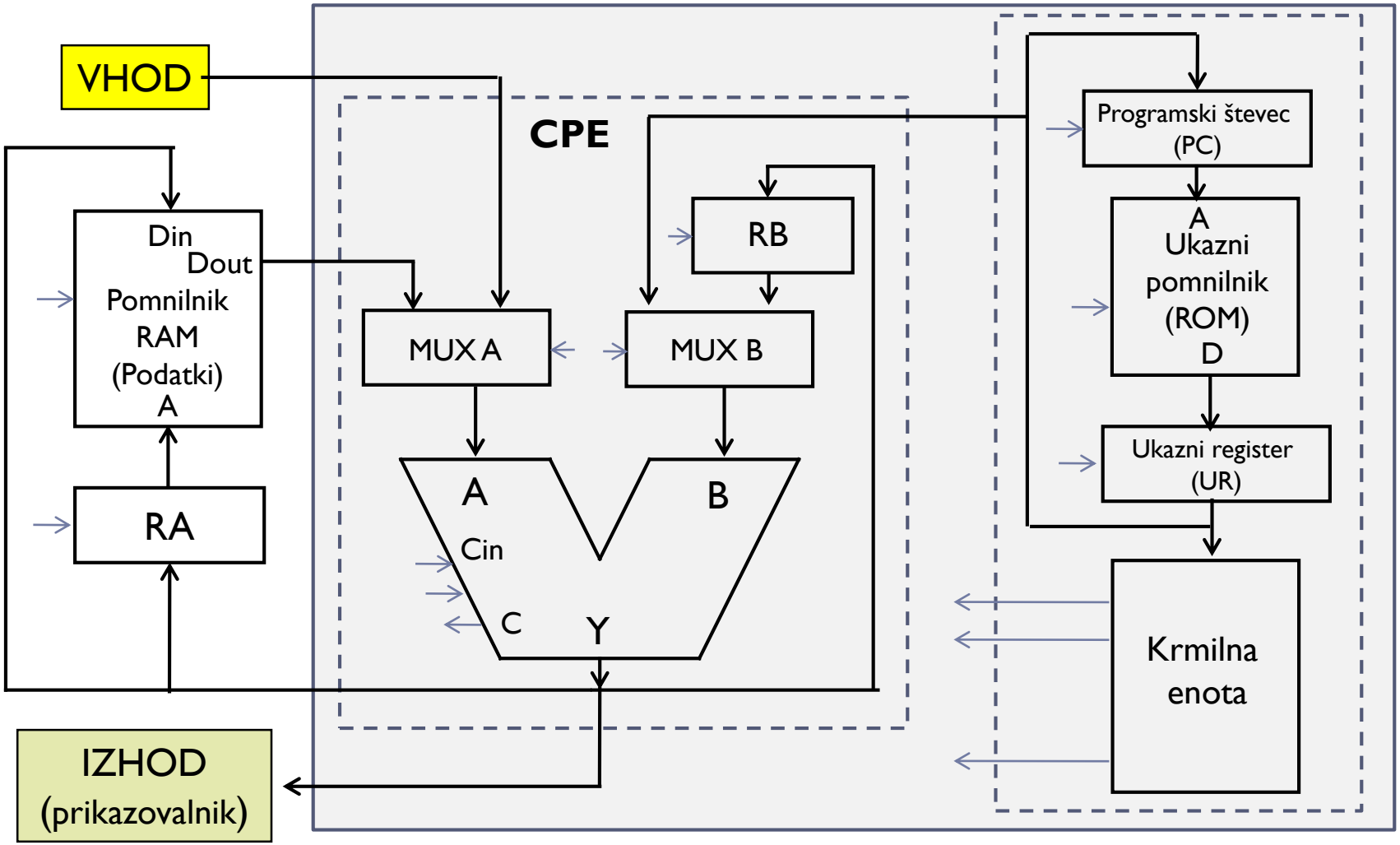
- Primer: 16 bitni procesor
  - Predstavitev delovanja - logisim
  - Nadgradnja procesorja z dodatnimi ukazi

- Postopek:
  1. Definicija: (a) arhitektura procesorja, (b) format ukazov in (c) nabor ukazov.
  2. Blok shema podatkovnih poti.
  3. Izbira gradnikov v logisimu in določitev krmilnih signalov.
  4. Logična shema procesorja s krmilnimi signali.
  5. Zapis krmilnih signalov v tabelo.
  6. Realizacija krmilne enote (logična vrata, ROM).
  7. Implementacija procesorja v logisimu.
  8. Testiranje procesorja za celoten nabor ukazov.

16-bitni procesor - Logisim

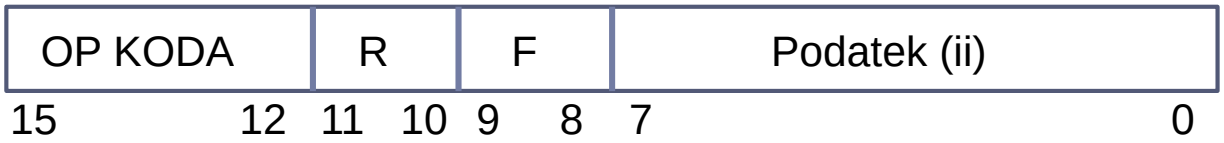


1a) Arhitektura 16-bitnega procesorja

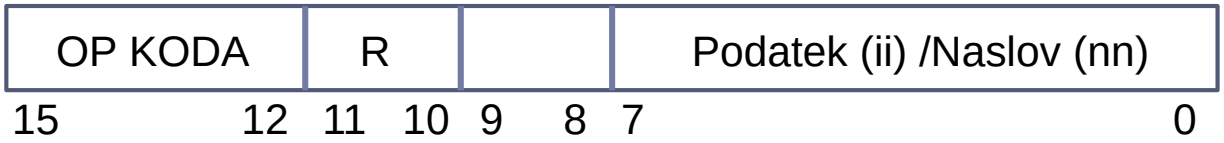


1b) Format ukazov (D-tip, P-tip)

D-tip: aritmetično /logične operacije nad podatki



P-tip: prenos podatkov in skočni ukazi



- Operacijska koda (OP KODA) - ukaz (4-biti)
- Podatek (D-tip) – takojšnji podatek v ukazu (8-bitov)
- Podatek/Naslov (P-tip) - takojšnji podatek ali naslov v ukazu (8-bitov)

R - registra RA in RB

R_1	R_0	Register
0	1	RA
1	0	RB
0	0	-

F - Funkcije ALE

F_1	F_0	Y	Zastavica
0	0	A+B	C
0	1	B - A	-
1	0	A & B	-
1	1	B	-

1c) Nabor ukazov

OP KODA	Ukaz	Funkcija	Tip
0001 (1)	ADD RB, Vhod	RB ← RB + Vhod	D-tip
0010 (2)	SUB RB, (RA)	RB ← RB - (RA)	D-tip
0011 (3)	INC RB	RB ← RB+1	D-tip
0100 (4)	MOVE R, #Pod	R ← SE(Podatek)	P-tip
0101 (5)	MOVE (RA), #Pod	(RA) ← SE(Podatek)	P-tip
0110 (6)	BRC Naslov	IF C=1, PC ←ZE(Naslov)	P-tip

Razširitev podatka, ki je v ukazu:

- SE – Sign Extend (raztegnitev predznaka)

  00011011 → **00000000** 00011011 (predznak = 0)

  10011011 → **11111111** 10011011 (predznak = 1)

- ZE – Zero Extend (raztegnitev ničle)

  00011011 → **00000000** 00011011

  10011011 → **00000000** 10011011

- Vsak ukaz predstavimo v dvojiškem in šestnajstiškem zapisu za določanje krmilne enote in vpis programa v ukazni pomnilnik.
- Vključimo tudi določanje prenosa Cin pri operaciji seštevanja v ALE.
- Prenos podatka na strani B izvedemo z definicijo vhoda Cin = 0

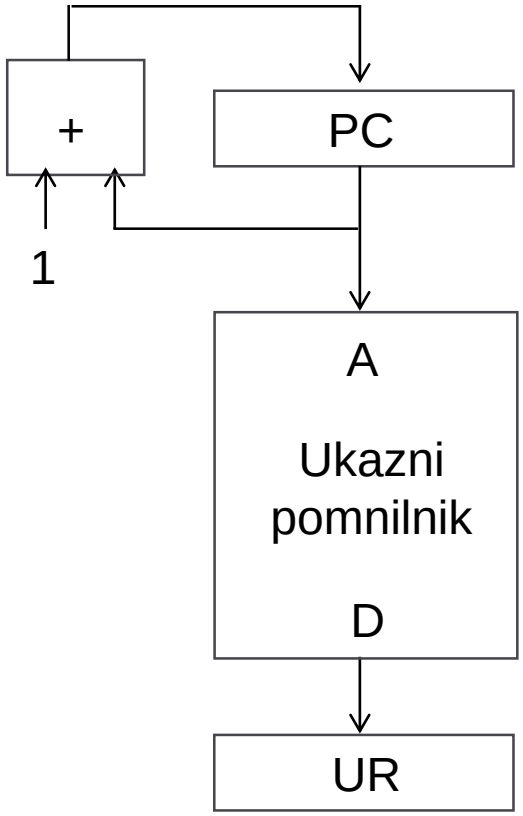
Ukaz	OP KODA (15-12)	R (R_1,R_0) (11,10)	F (F_1,F_0) (9,8)	Cin	Ukaz (b_{15}-b_8)	Ukaz (Hex)
ADD RB,Vhod	0001	RB (10)	A + B (00)	0	0001 10 00	18 xx
SUB RB, (RA)	0010	RB (10)	B – A (01)	0	0010 10 01	29 xx
INC RB	0011	RB (10)	B + 1 (11)	1	0011 10 11	3b xx
MOVE RA, #Pod	0100	RA (01)	B (11)	0	0100 01 11	47 ii
MOVE RB, #Pod	0100	RB (10)	B (11)	0	0100 10 11	4b ii
MOVE (RA), #Pod	0101	- (00)	B (11)	0	0101 00 11	53 ii
BRC Naslov	0110	- (00)	-	-	0110 00 00	60 nn

xx – karkoli (0 ali 1)
ii – podatek
nn - naslov

Podatkovne poti

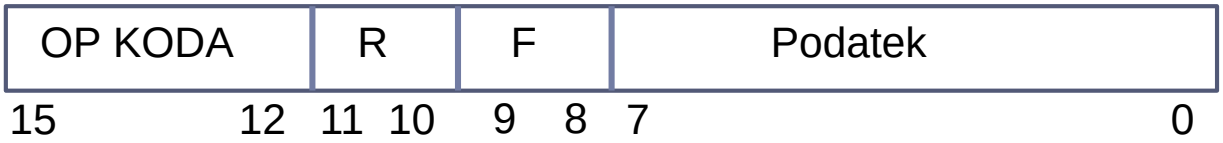
Dostava ukaza:

1. Števec naslovi ukazni pomnilnik, ukaz se pojavi na izhodu D in se shrani v Ukazni register (UR), programski števec (PC) se poveča za 1.
2. Ukaz se dekodira in na izhodu krmilne enote se pojavijo krmilni signali, ki zagotavljajo dostavo in zvajanje ukaza.

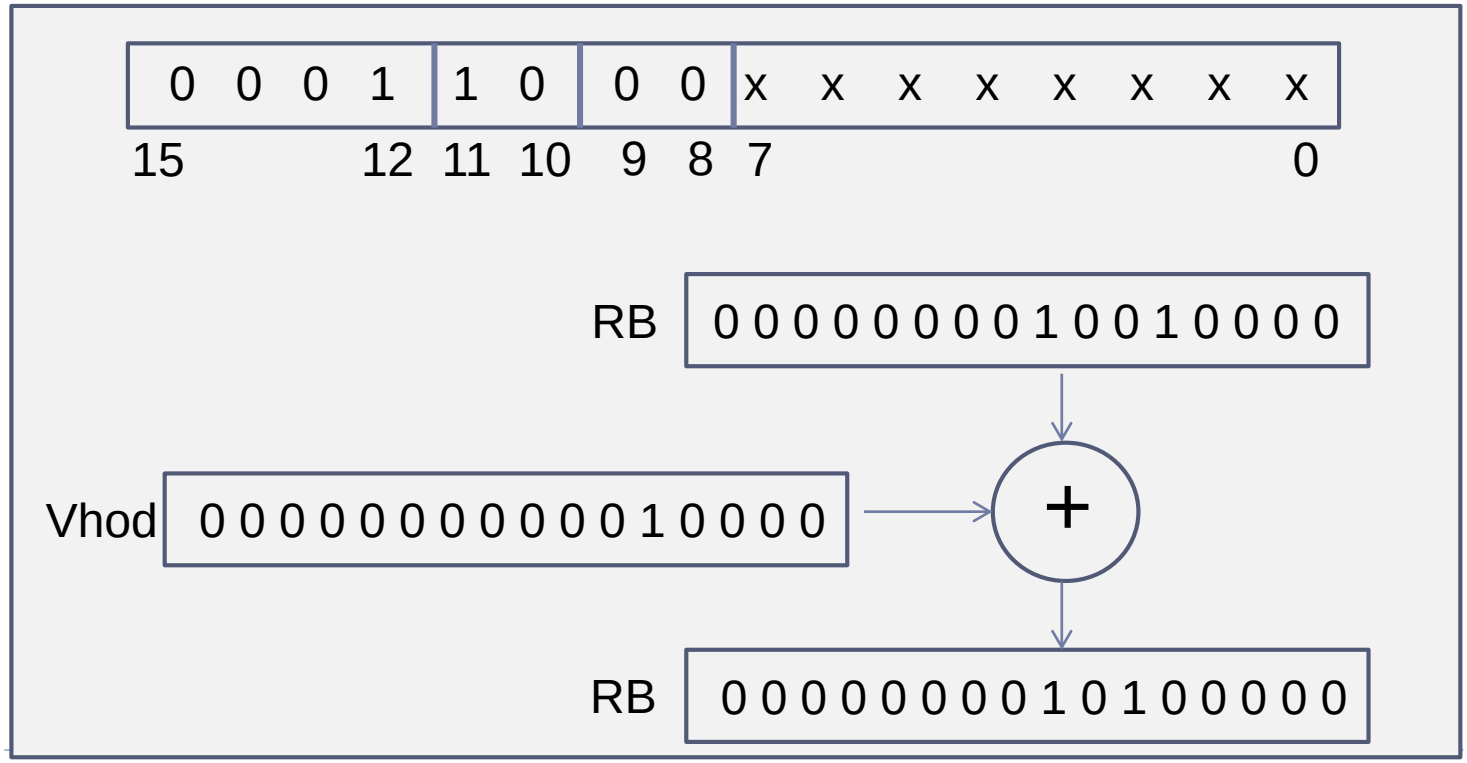


Dostava ukaza:
UR <= Pom[PC]
PC <= PC+ 1

ADD RB ,Vhod (D-tip RB ← RB + Vhod)

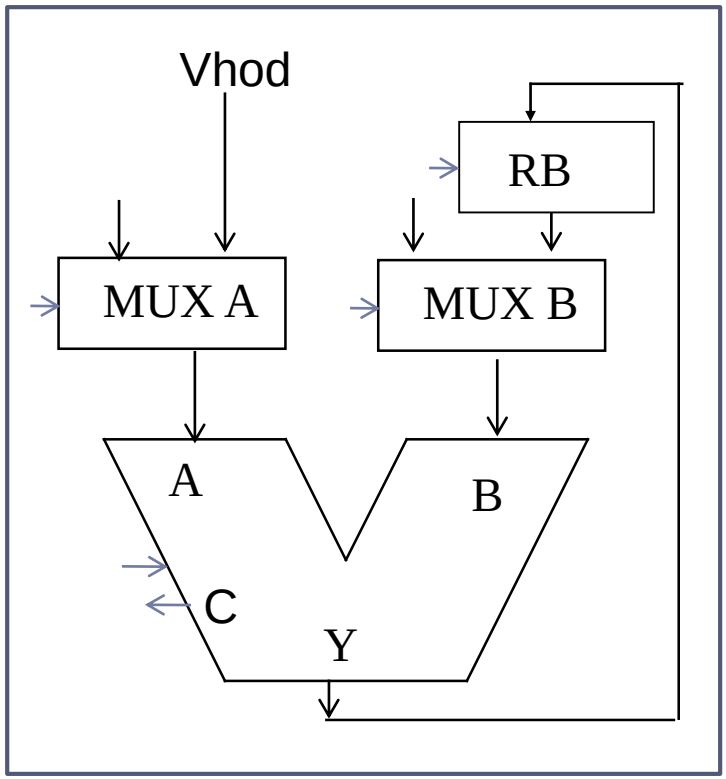


Seštevanje vsebine registra RB z vhodnim podatkom (Vhod) in zapis rezultata v register RB.



Podatkovne poti:

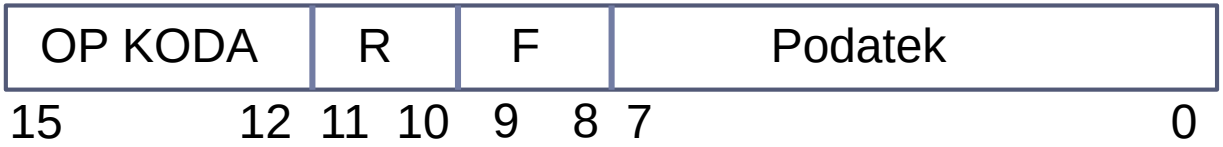
- Nastaviti podatek (16_{10}) na vhodu MUX-a za vhod A (ALE).
- Poslati vsebino registra (RB) na vhod MUX-a za vhod B (ALE).
- Sešteti A+B in poslati vsoto na izhod ALE.
- Podatek shraniti v register RB.



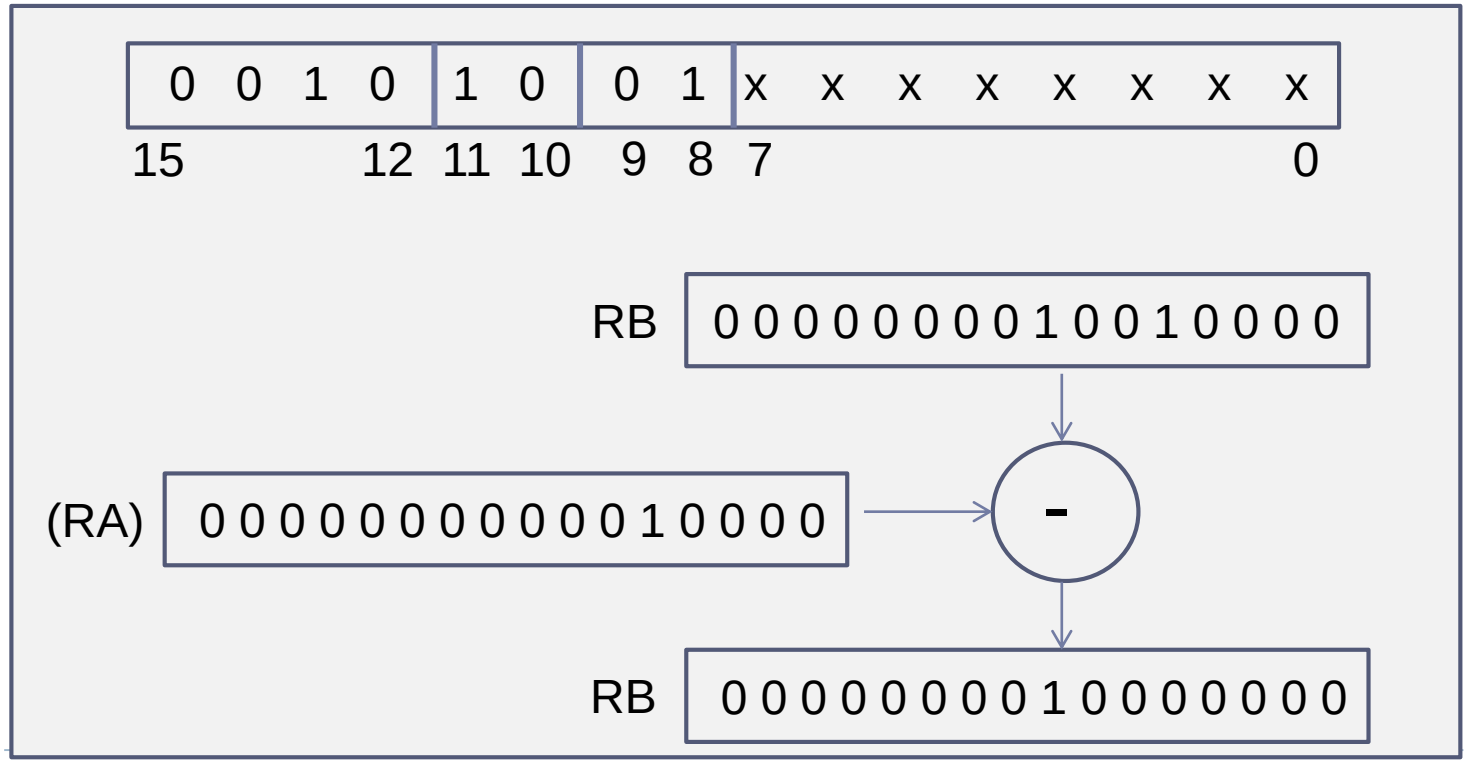
ADD RB, Vhod

A <= Vhod
B <= RB
Y <= A+B
RB <= Y

SUB RB, (RA) (D-tip RB ← RB – (RA))

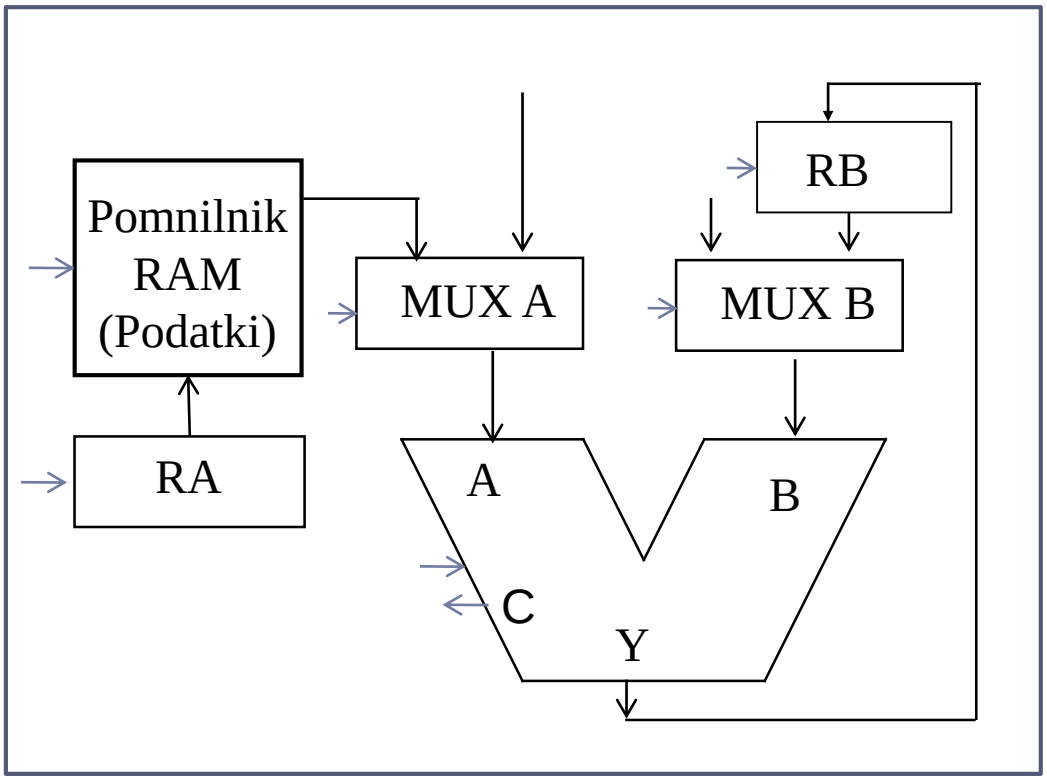


Odštevanje pomnilne besede, ki je na naslovu podanem v registru RA od vsebine registra RB in zapis rezultata v register RB.



Podatkovne poti:

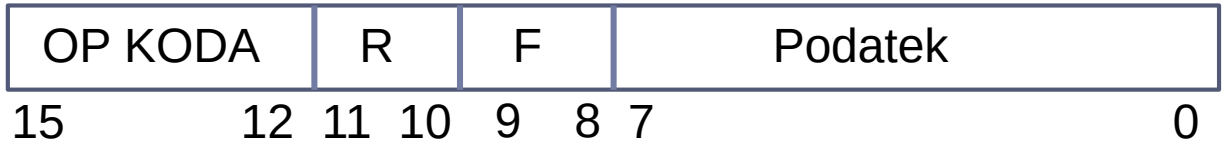
- Iz pomnilnika ROM poslati vsebino besede na naslovu iz RA na vhod MUX-a za vhod A (ALE).
- Poslati vsebino registra (RB) na vhod MUX-a za vhod B (ALE).
- Odšteti B od A in poslati razliko na izhod ALE.
- Podatek shraniti v register (RB).



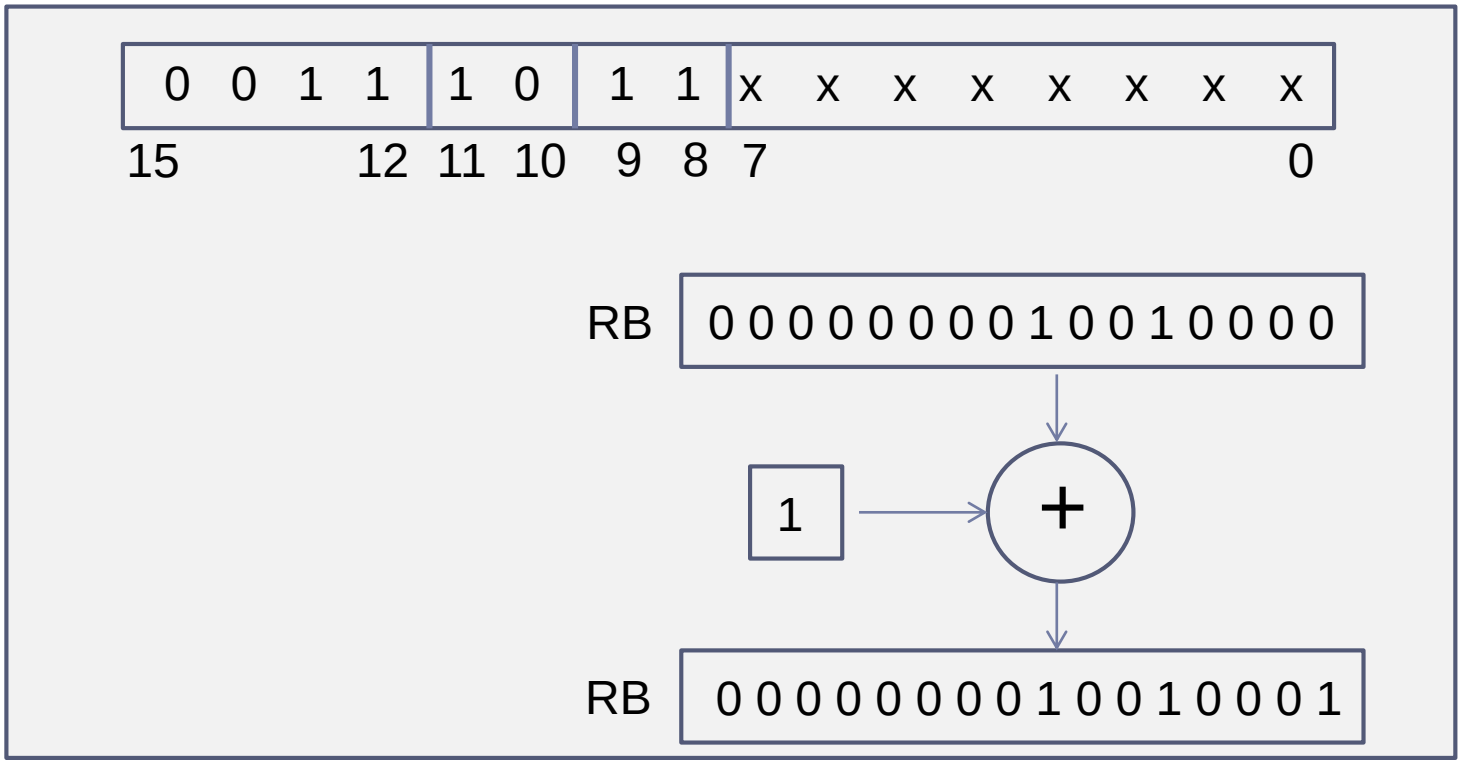
SUB RB, (RA)

A <= P(RA)
B <= RB
Y <= B - A
RB <= Y

INC RB (D-tip RB ← RB + 1)

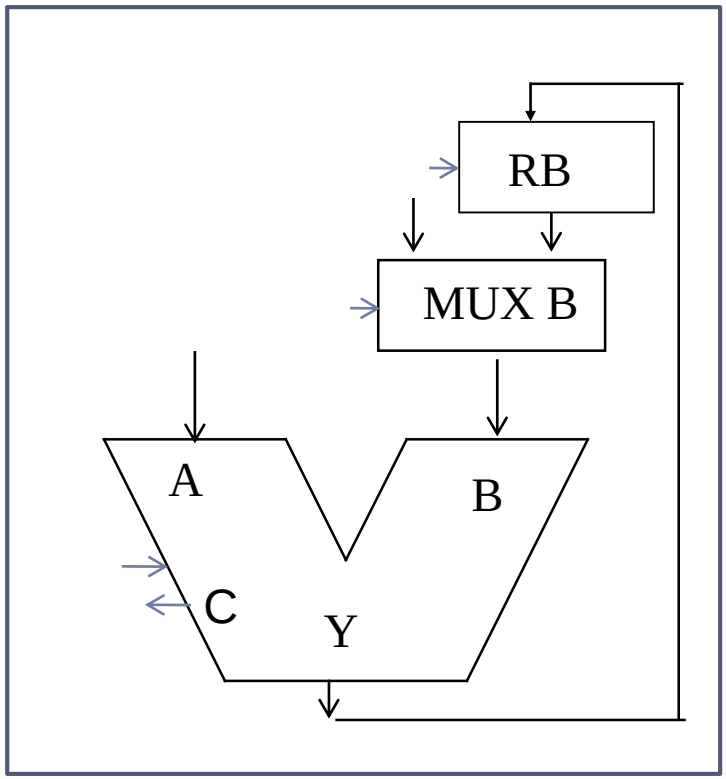


Inkrement registra RB za 1 in zapis rezultata v register RB.



Podatkovne poti:

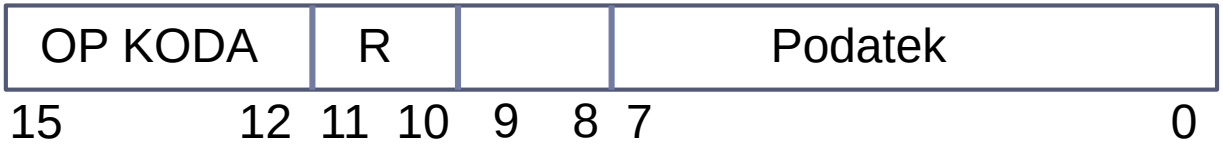
- Poslati vsebino registra (RB) na vhod MUX-a za vhod B (ALE).
- Prišteti B vrednost 1 in poslati vsoto na izhod ALE.
- Podatek shraniti v register (RB).



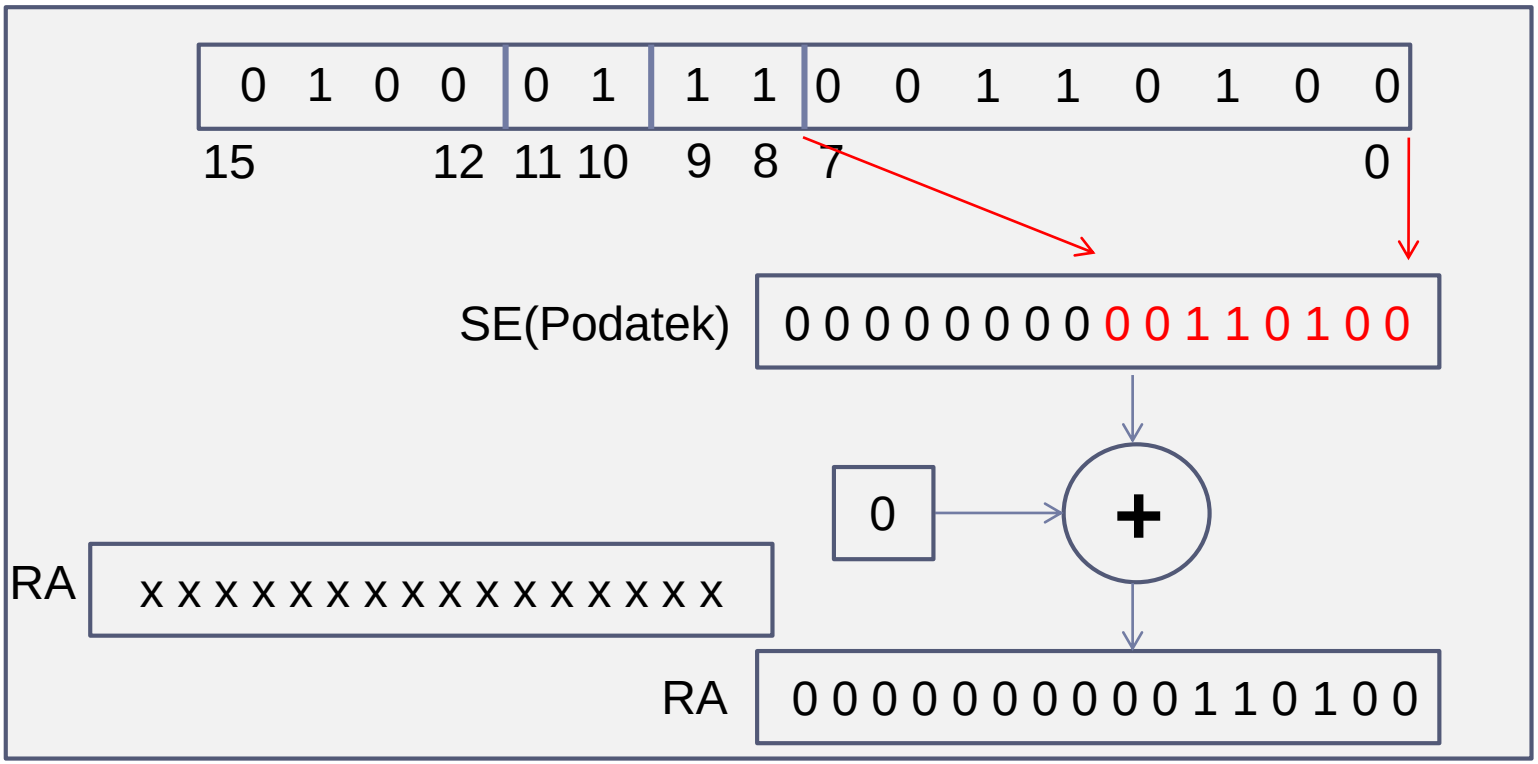
INC RB

B <= RB
Y <= B + 1
RB <= Y

MOVE RA, #Pod (P-tip RA← SE(Podatek))

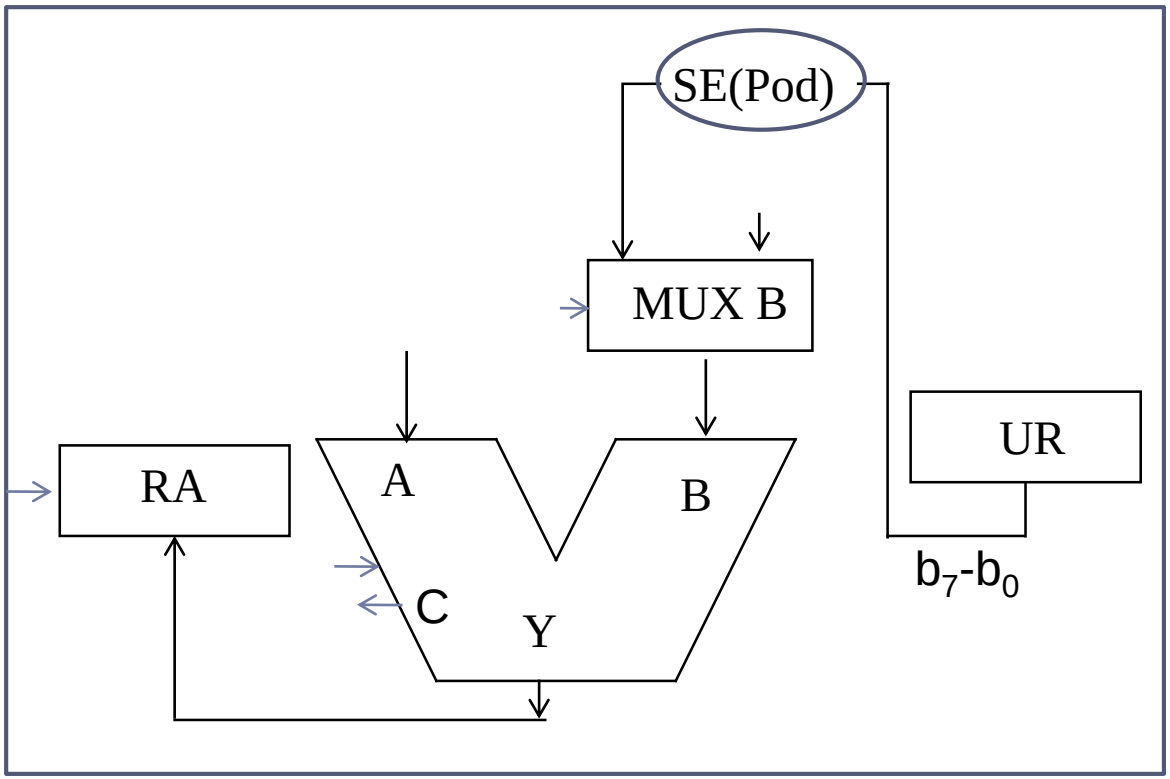


S predznakom razširjen takojšnji operand se shrani v register RA.



Podatkovne poti:

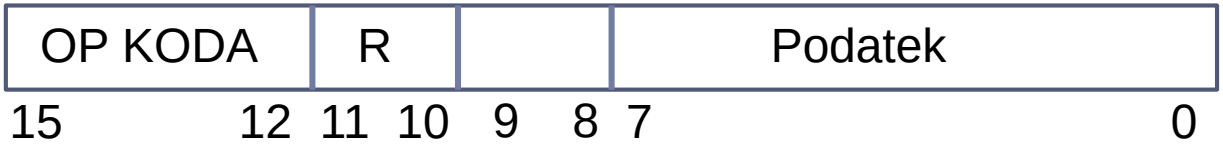
- Poslati s predznakom razširjen takojšnji operand na vhod B (ALE).
- Podatek na B poslati na izhod ALE.
- Podatek shraniti v register RA.



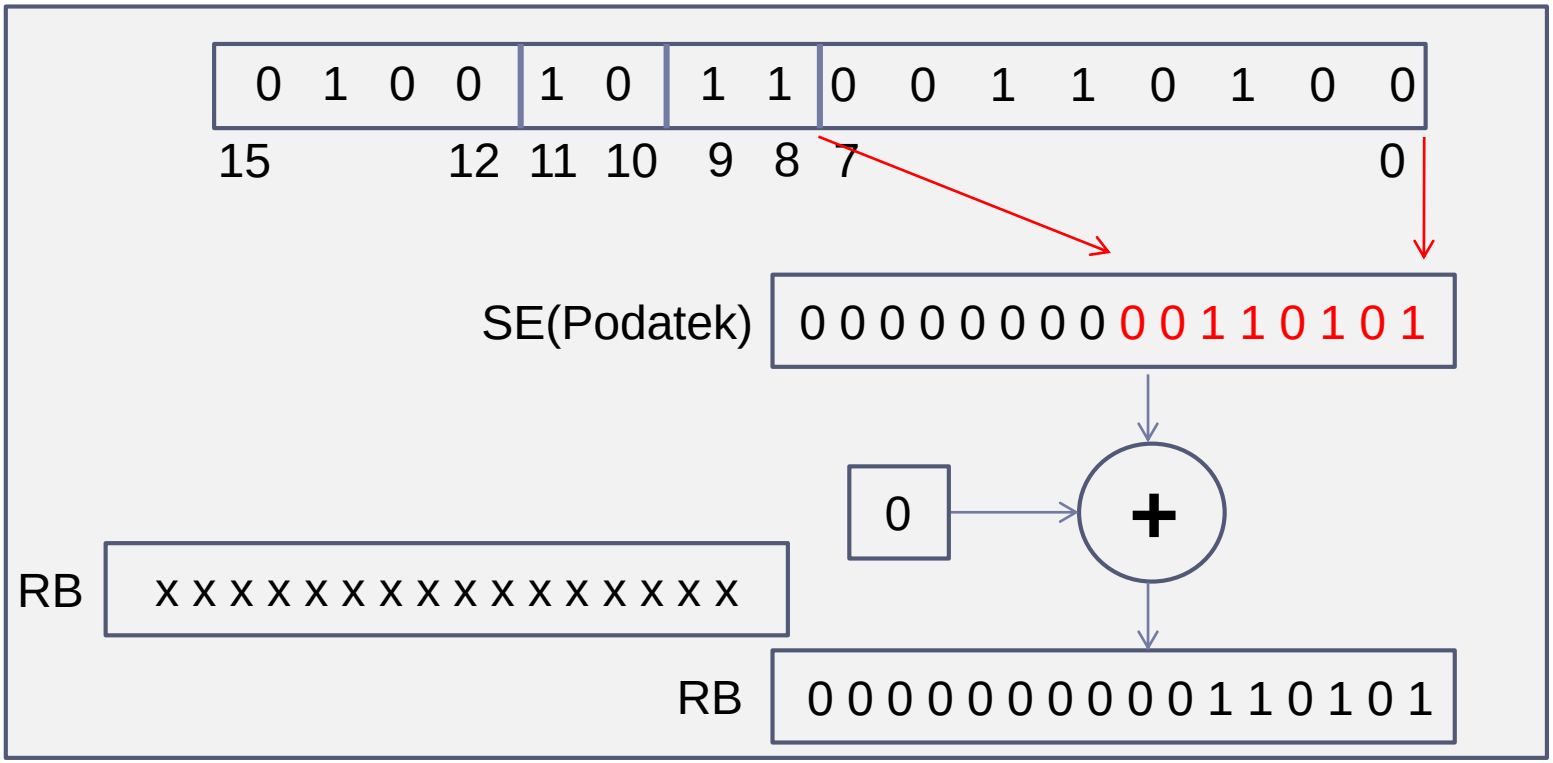
MOVE RA, #Pod

B <= SE(Pod)
Y <= B
RA <= Y

MOVE RB, #Pod (P-tip RB← SE(Podatek))

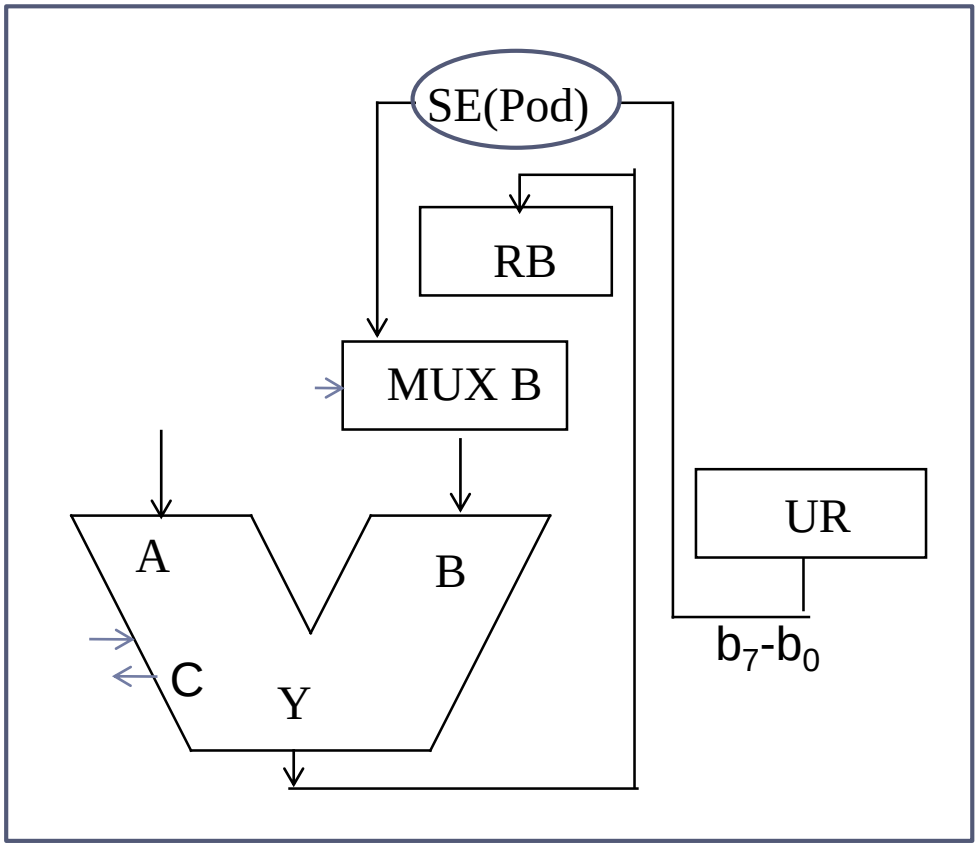


S predznakom razširjen takojšnji operand se shrani v register RB.



Podatkovne poti:

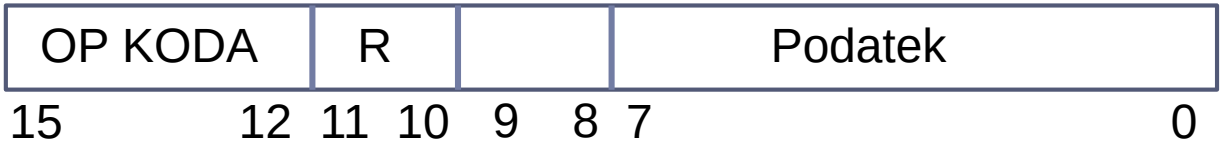
- Poslati s predznakom razširjen takojšnji operand na vhod B (ALE).
- Podatek na B poslati na izhod ALE.
- Podatek shraniti v register RB.



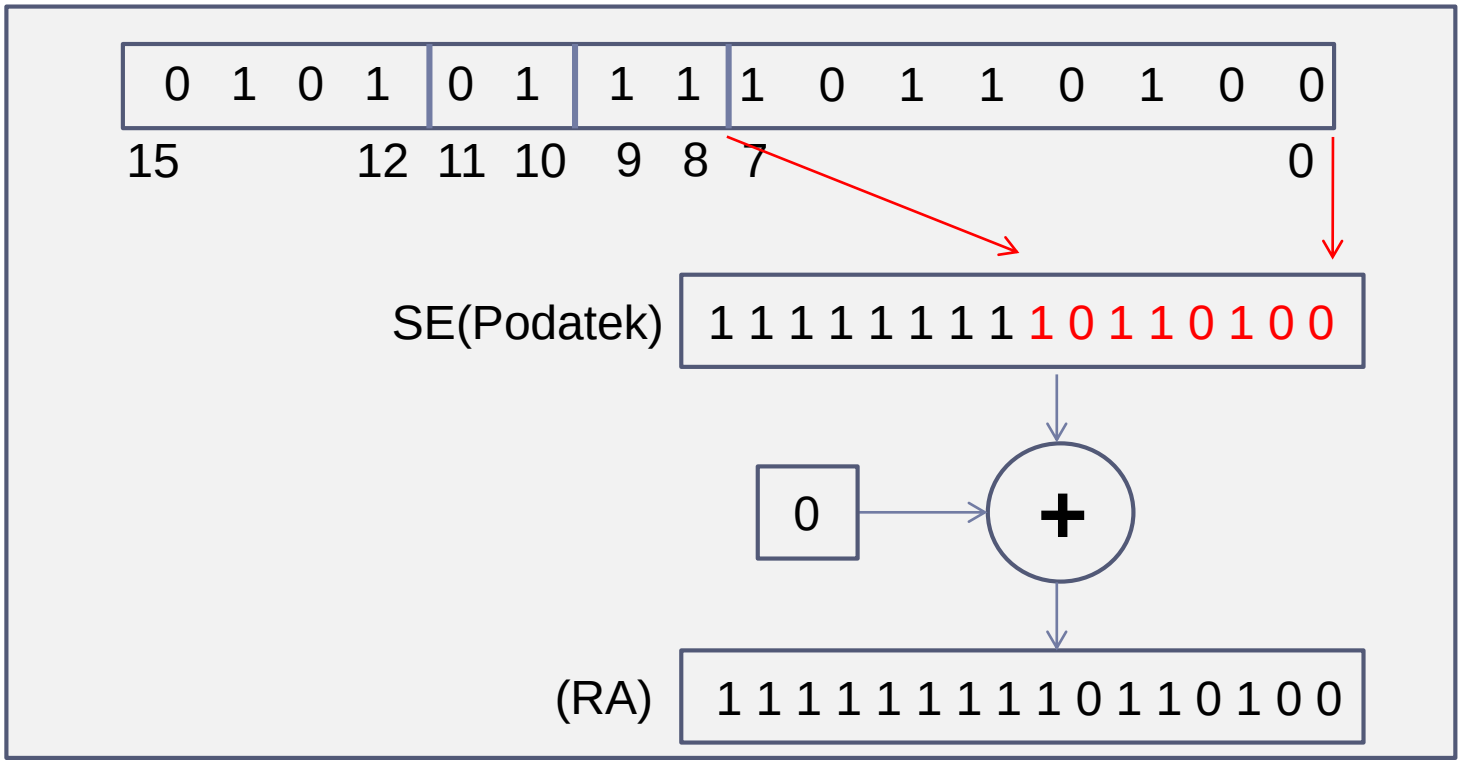
MOVE RB, #Pod

B <= SE(Pod)
Y <= B
RB <= Y

MOVE (RA), #Pod (P-tip (RA)← SE(Podatek))

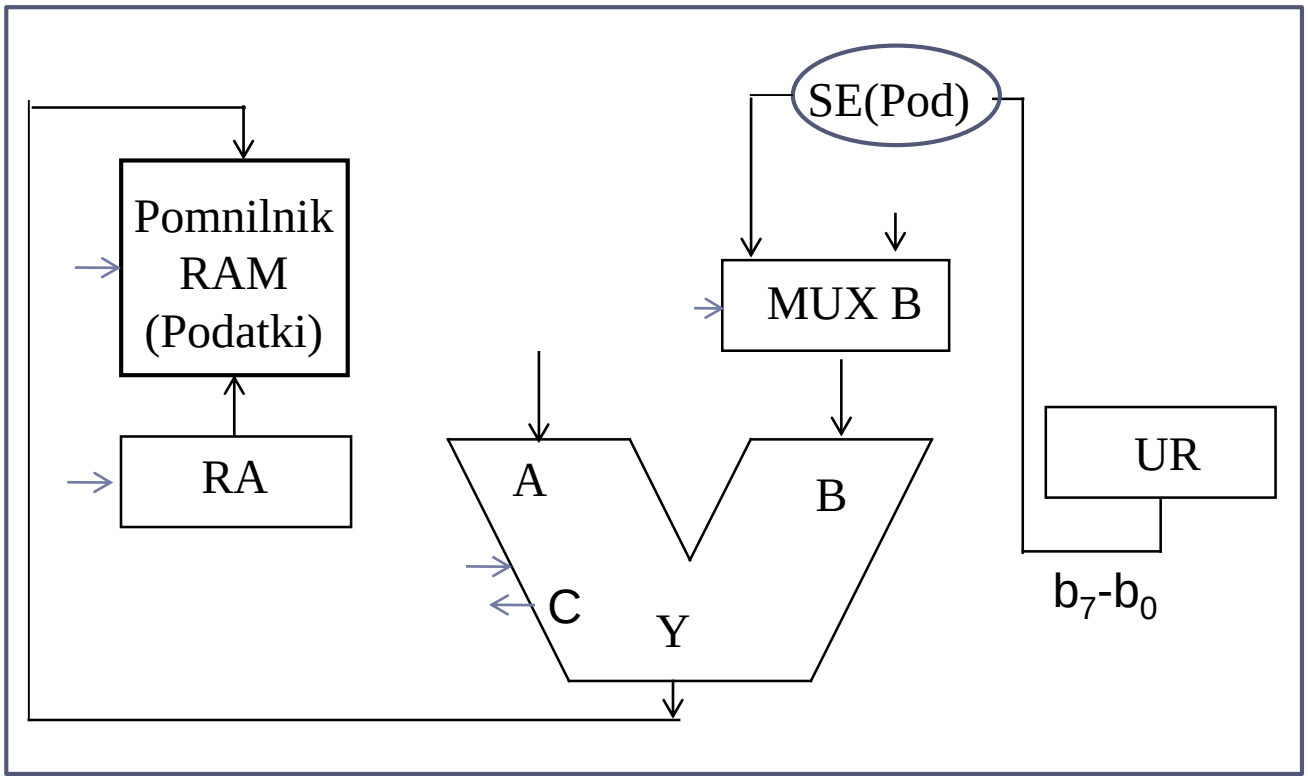


S predznakom razširjen podatek se shrani v pomnilnik na naslov, ki se nahaja v registru RA.



Podatkovne poti:

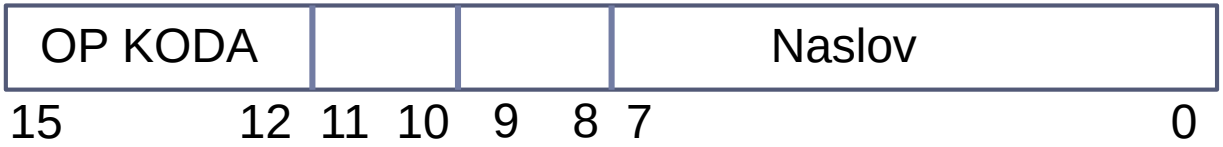
- Poslati s predznakom razširjen takojšnji operand (Pod) na vhod B (ALE).
- Podatek na B poslati na izhod ALE .
- Podatek shraniti v pomnilnik na naslov, ki je v registru RA.



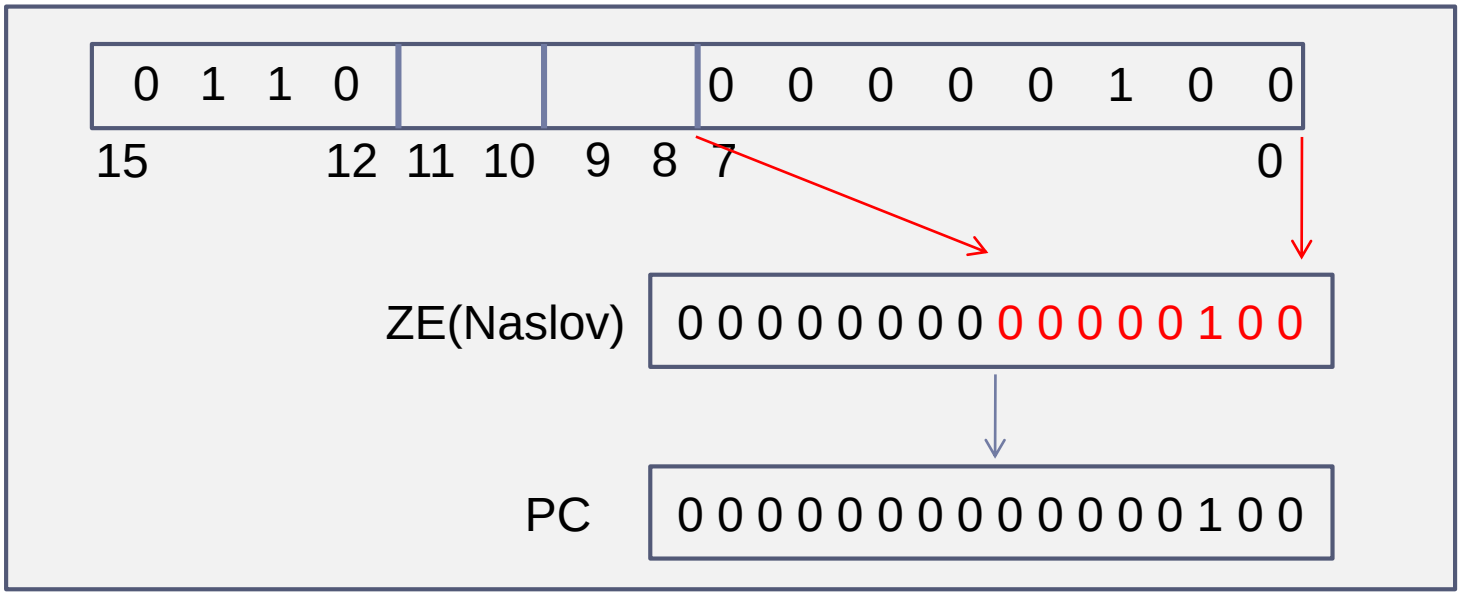
MOVE (RA), #Pod

B <= SE(Pod)
Y <= B
Pom(RA) <= Y

BRC Naslov (P-tip PC← Naslov)

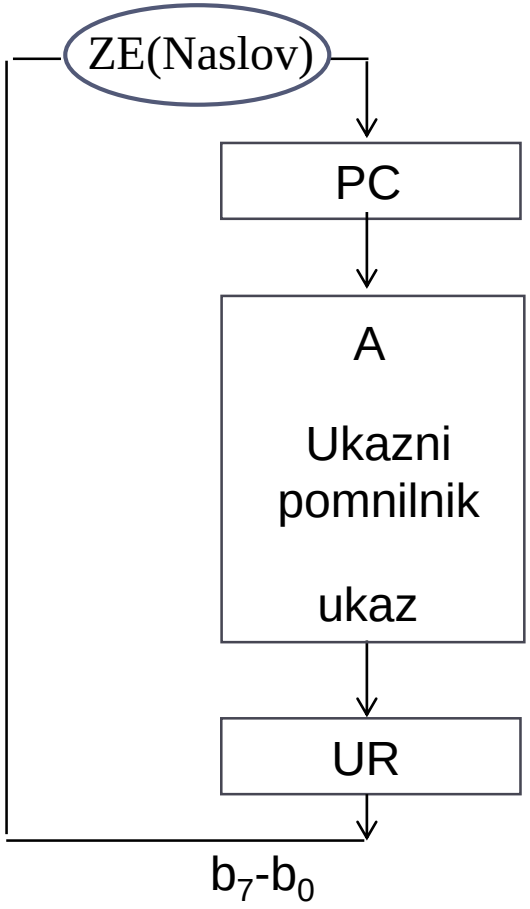


Z ničlami razširjeni naslov se shrani v PC.



Podatkovne poti:

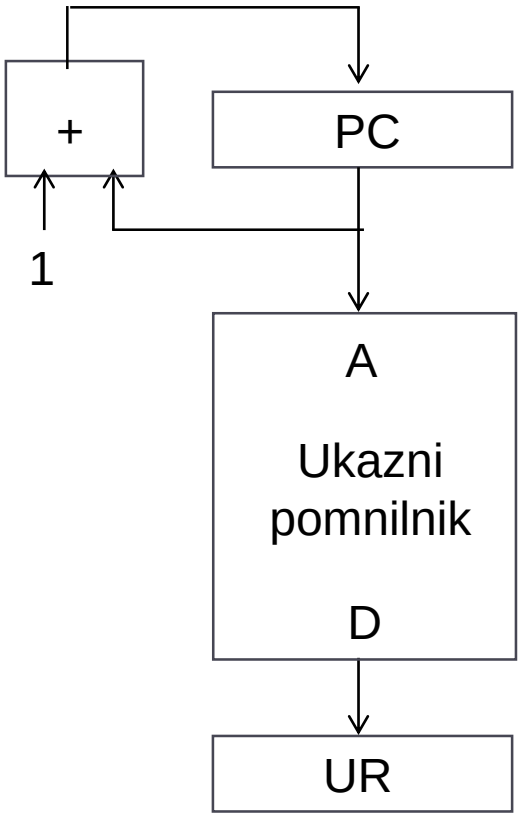
- Poslati z ničlo razširjen naslov na vhod registra PC.



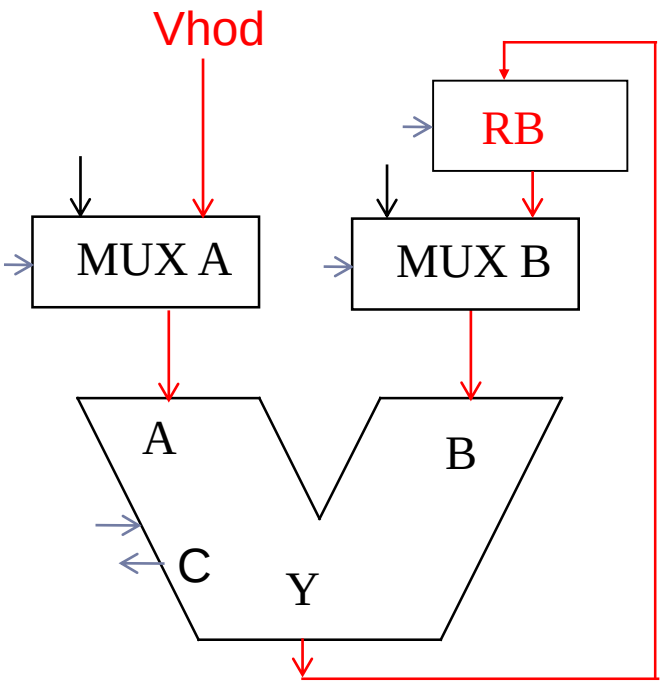
BRC Naslov

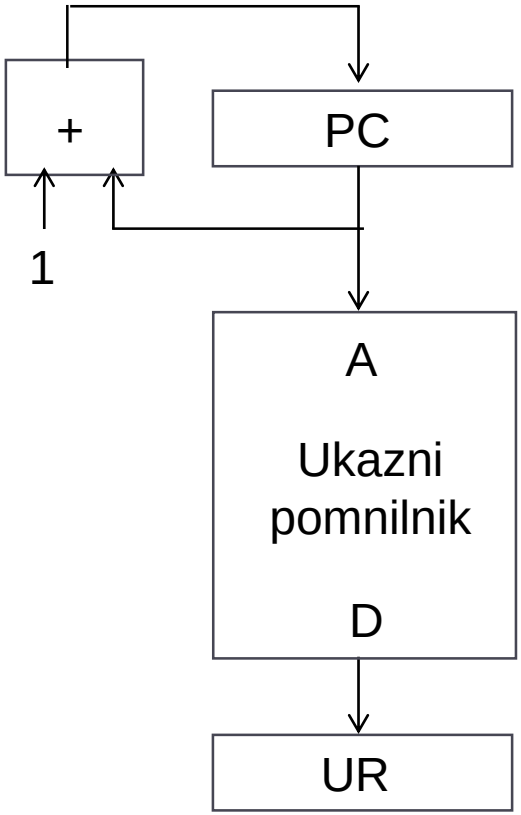
PC <= ZE(Naslov)

Povezovanje podatkovnih poti v shemo procesorja
dostava

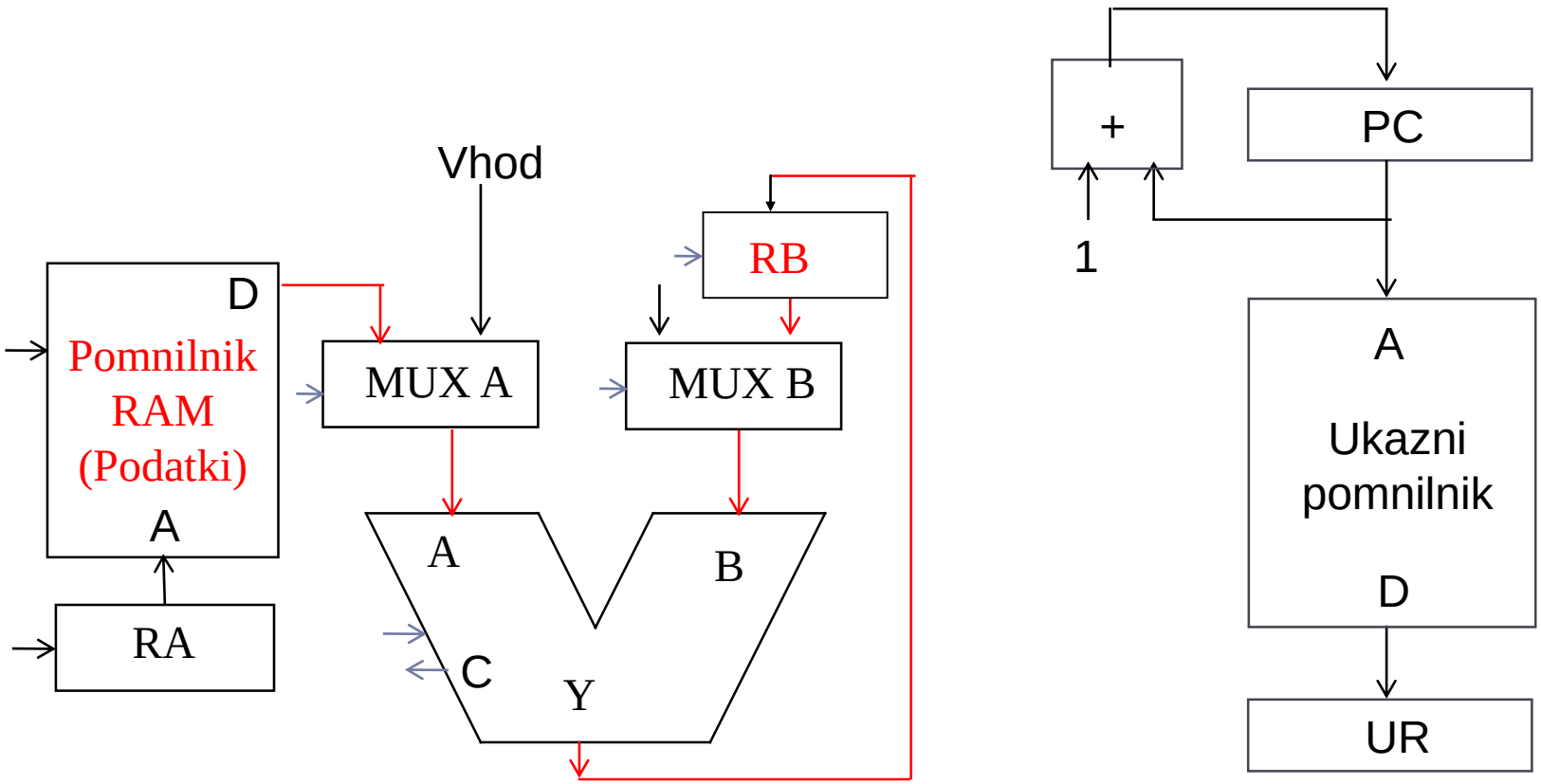


dostava, add

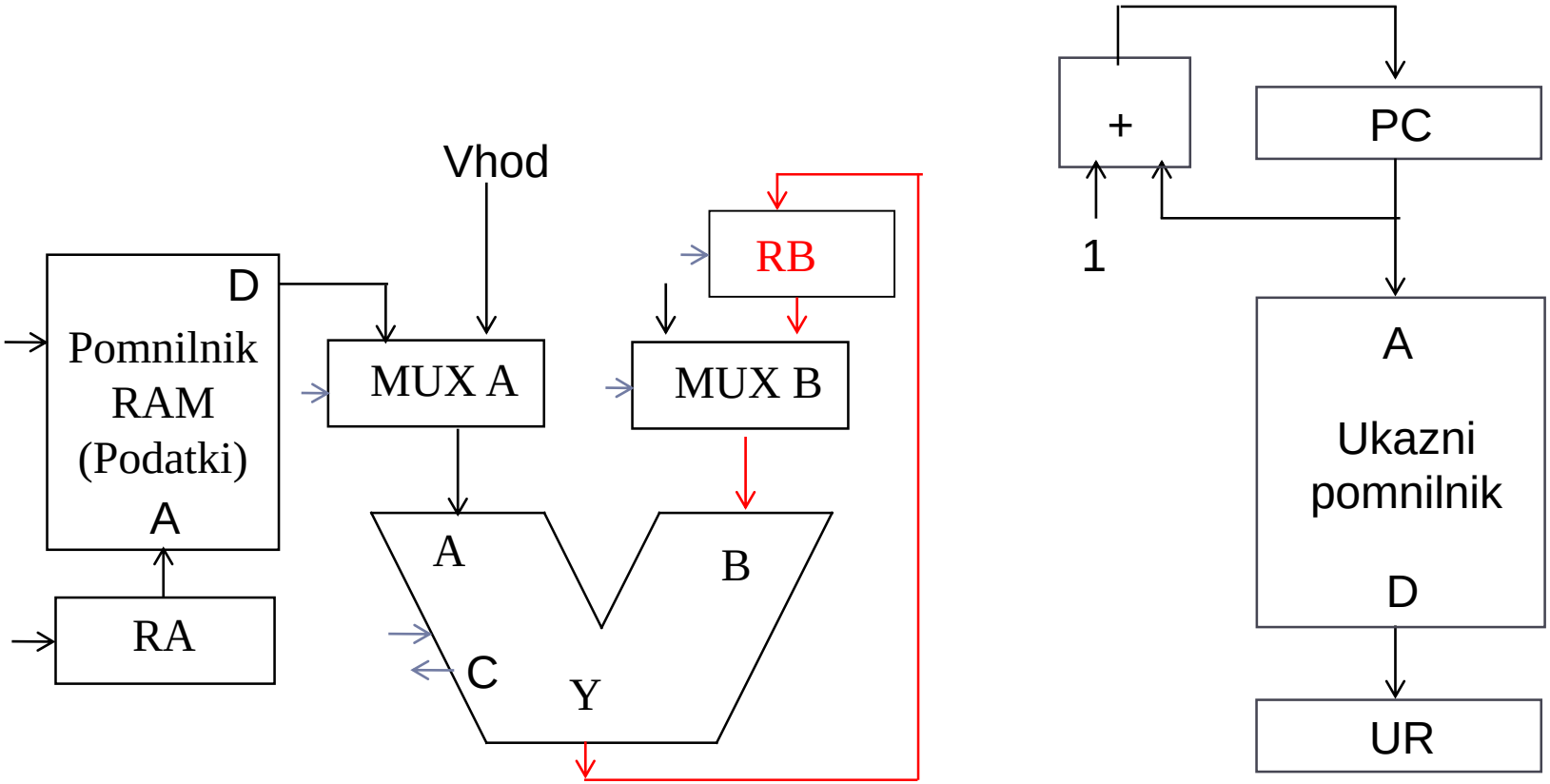




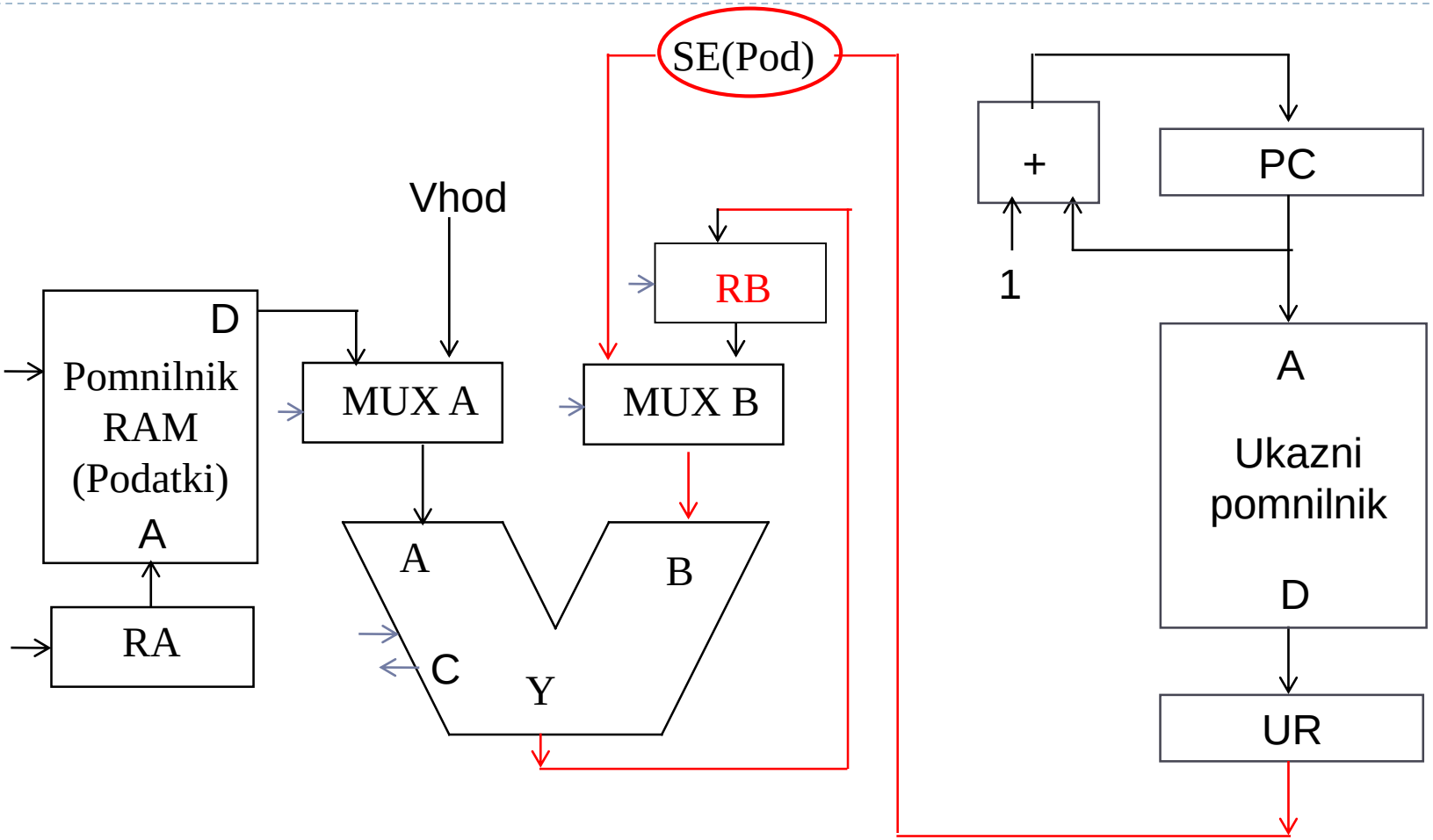
dostava, add, sub



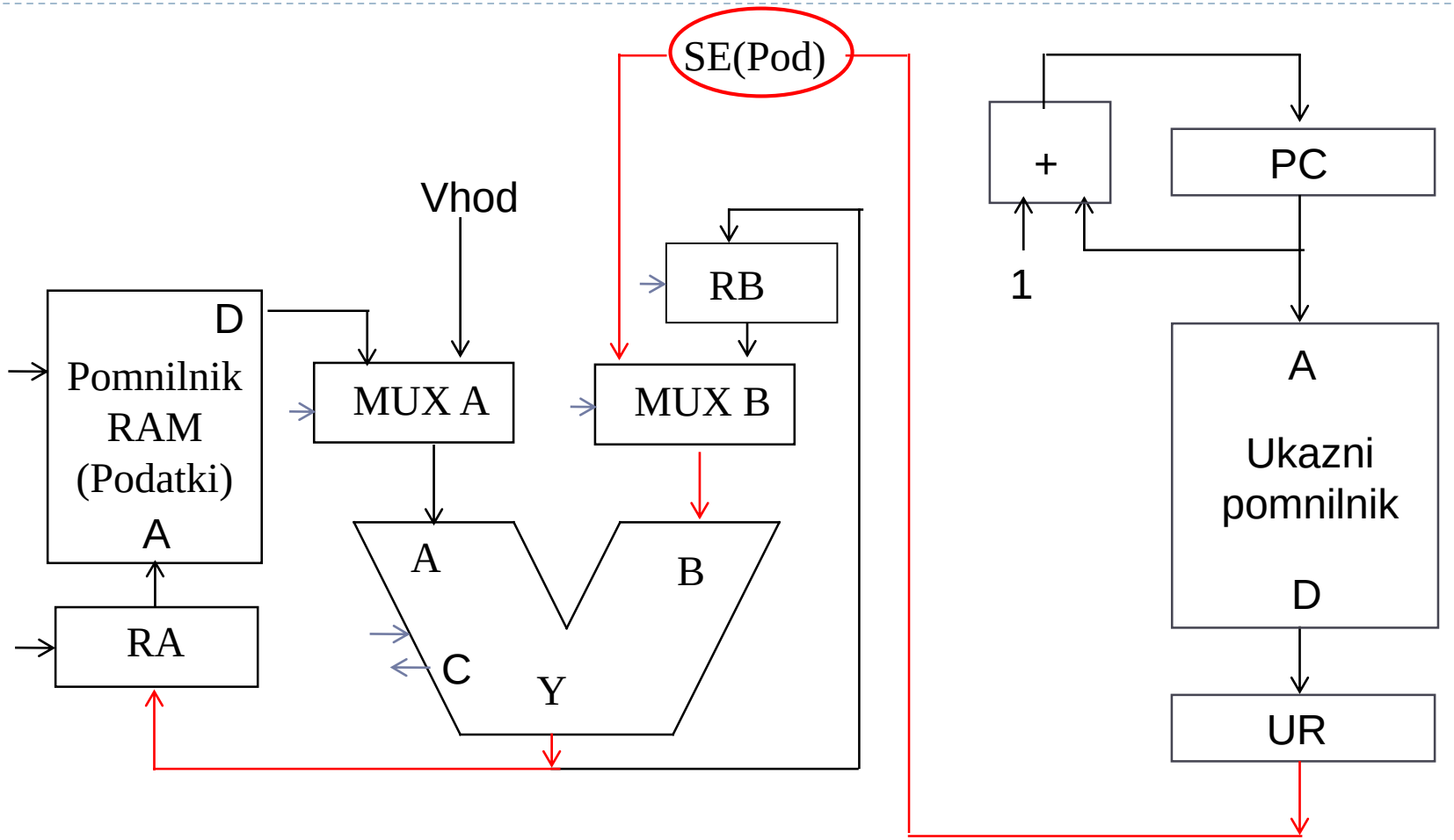
dostava, add, sub, inc



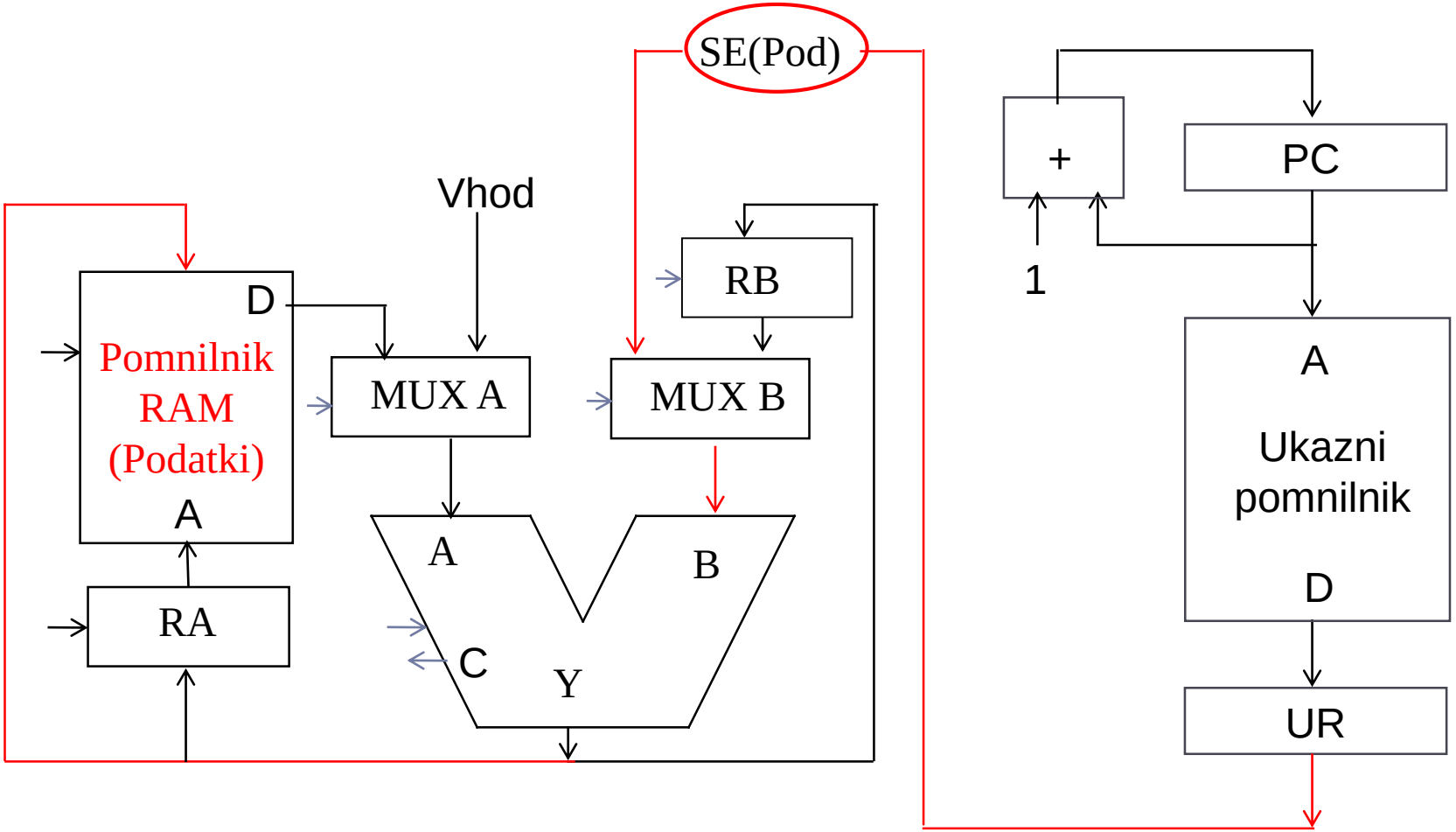
dostava, add, sub, ink, move (RB)



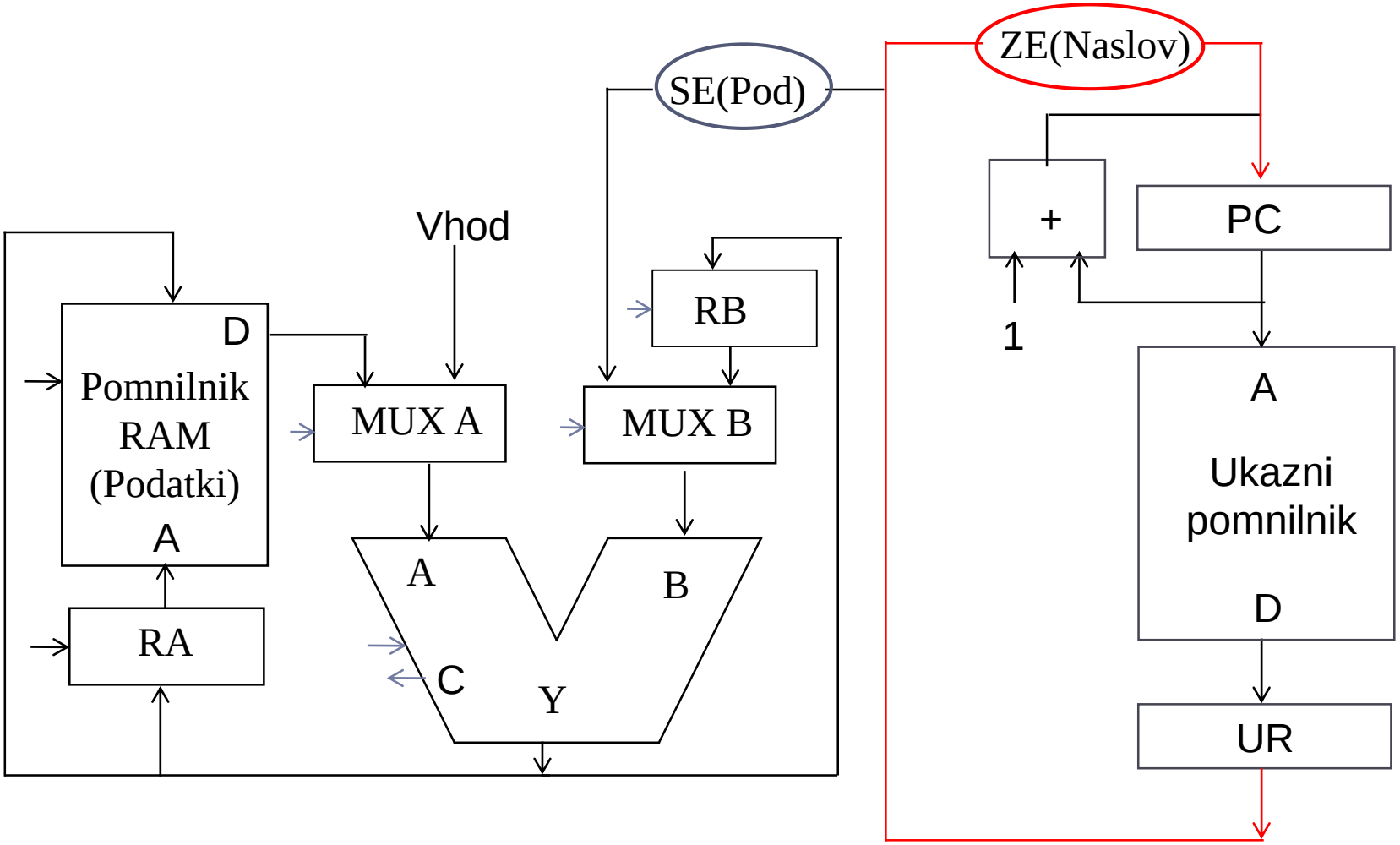
dostava, add, sub, ink, move (RB), move (RA)



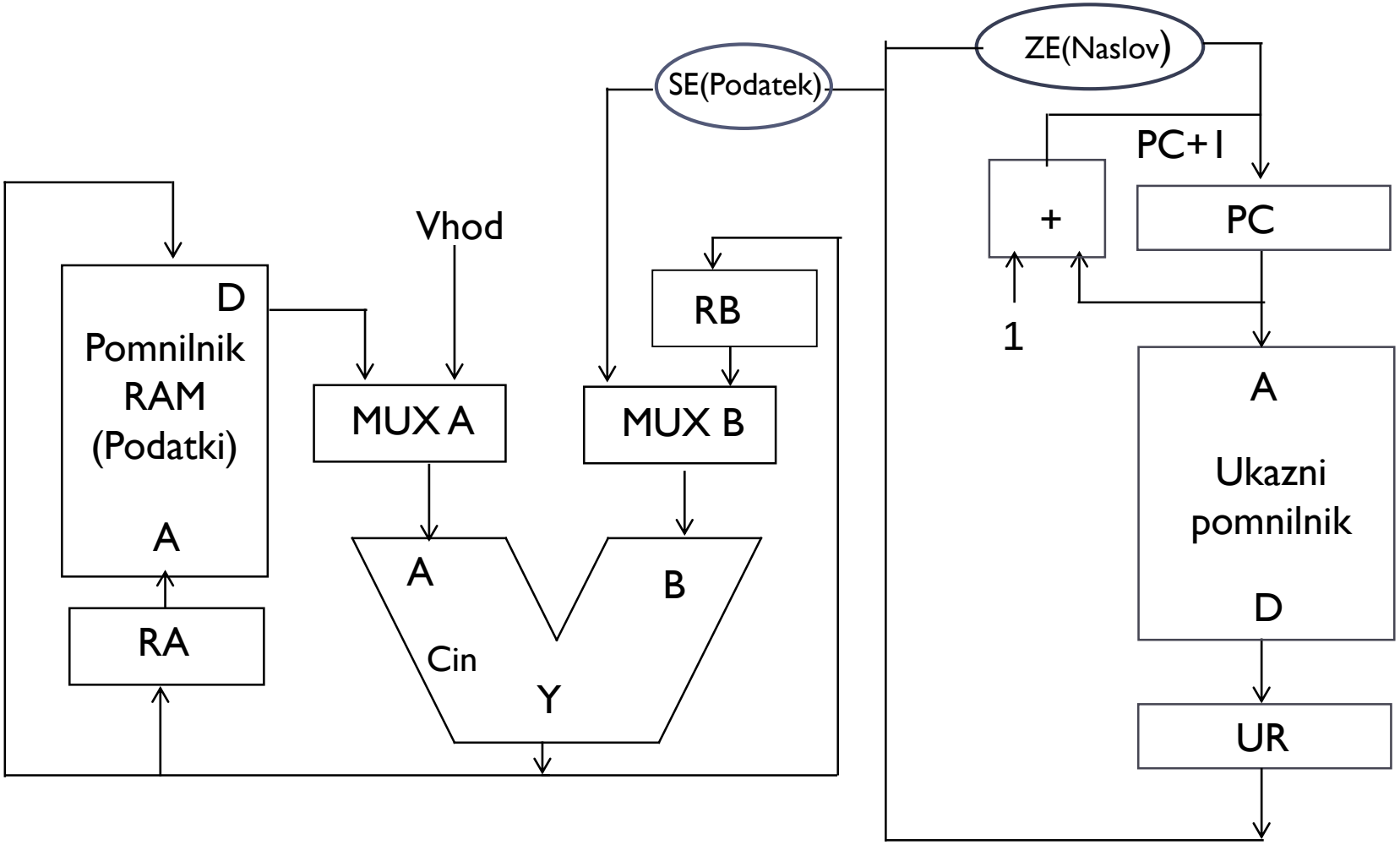
dostava, add, sub, ink, move (RB), move (RA), move (RAM)



dostava, add, sub, ink, move (RB), move (RA), move (RAM), BRC

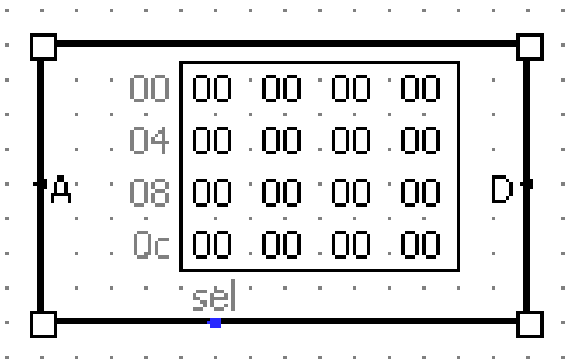


2- Blok shema podatkovnih poti



3- a)Gradniki v logisimu

- Pomnilnik:
  - ROM - Ukazni pomnilnik (ukazi)
  - RAM - Podatkovni pomnilnik (podatki)
- Programski števec (PC)
- Registri:
  - Ukazni register (UR)
  - Register RA
  - Register RB
- Multiplekser (MUX A, MUX B)
- Aritmetično logična enota (ALE)
- Inkrementer (seštevalnik)
- Vezje za raztegnitev predznaka (SE) in raztegnitev ničle (ZE)



Ukazni pomnilnik:

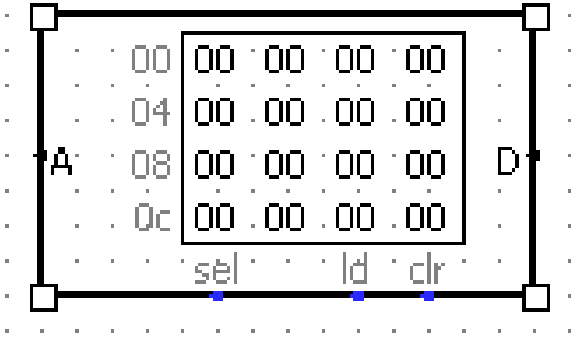
Naslovni vhodi (A- address)

Izhodi (D- Data)

sel (cs- chip select):

0 – onemogoča delovanje

1 – omogoča delovanje



sel = 1

Podatkovni pomnilnik:

Naslovni vhodi (A- address)

Vhodi/Izhodi (D- Data)

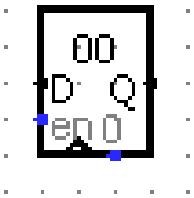
sel (cs- chip select)

ld - Load (Podatki na izhod – r/w)

0 – vpis (vhod)

1- branje (izhod)

clr - Clear (tipka za brisanje vsebine)

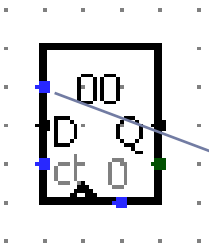


Register:

Vhod (D-Data) , Izhod (Q-Output)

en- Enable – omogoči vpis z uro (1- vpis v register)

clr- Clear (tipka za brisanje vsebine)



Števec:

Vhod (D-Data) , Izhod (Q-Output)

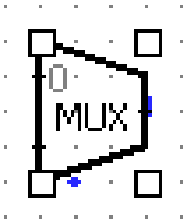
ct - count – inkrement ali vpis

Load – vpis

clr- Clear (tipka za brisanje vsebine)

ct = 1 in Load = 0 (povečevanje števca za 1)

ct = 0 in Load = 1 (vpis v števec)



1- naslovni multiplekser:

Naslovni vhod (A_0)

Podatkovna vhoda (I_0, I_1) in Izhod

ALE - Aritmetično logična enota

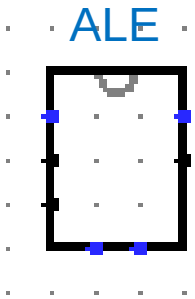
Vhodni podatki: A, B

Izhod: Y

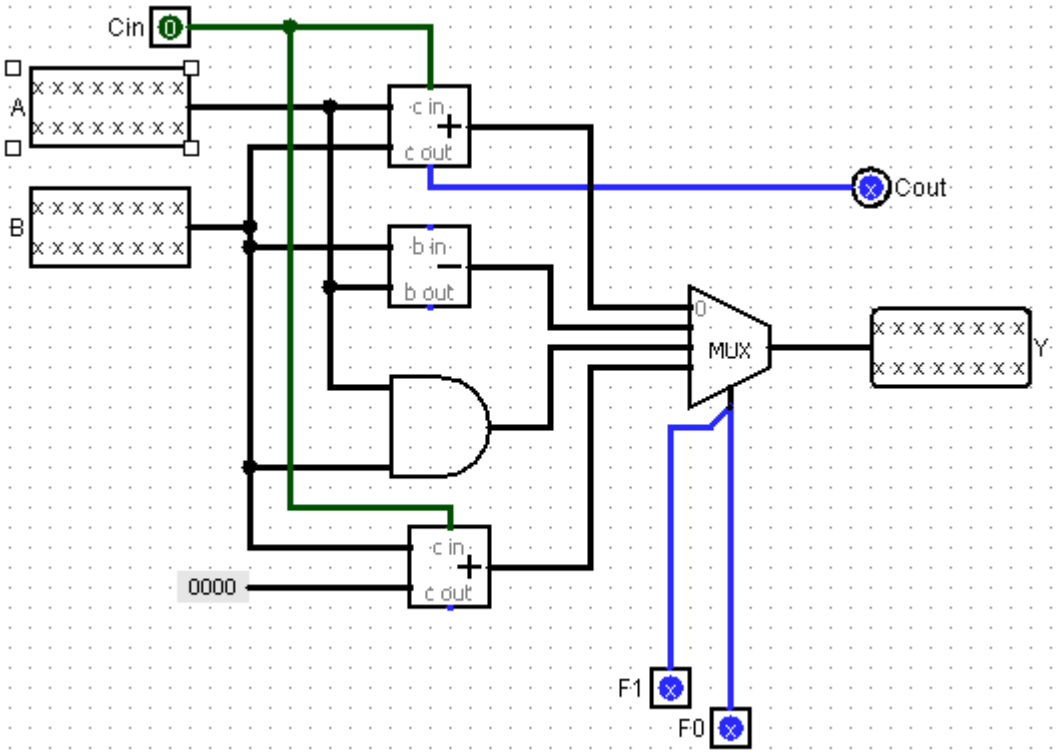
Cin – prenos na vhodu

Cout – prenos kot rezultat seštevanja (A+B)

Gradnik izdelan v logisimu



F1	F0	Y	Opis
0	0	A+B	Seštevanje
0	1	B - A	Odštevanje
1	0	A&B	Konjunkcija
1	1	B	Seštevanje (B+0),Cin =0



3 – b) Definicija krmilnih signalov

Register RA: **ra** (1 bit)
0: ni vpisa
1: vpis

Register RB: **rb** (1 bit)
0: ni vpisa
1: vpis

MUX A: **ma** (1 bit)
0: Vhod
1: (RA)

MUX B: **mb** (1 bit)
0: RB
1: SE(imm)

Prenos Cin (1 bit): 0, 1

Funkcije ALE: $\mathbf{F_1,F_0}$ (2 bita)
00: A+B
01: B-A
10: A&B
11: B

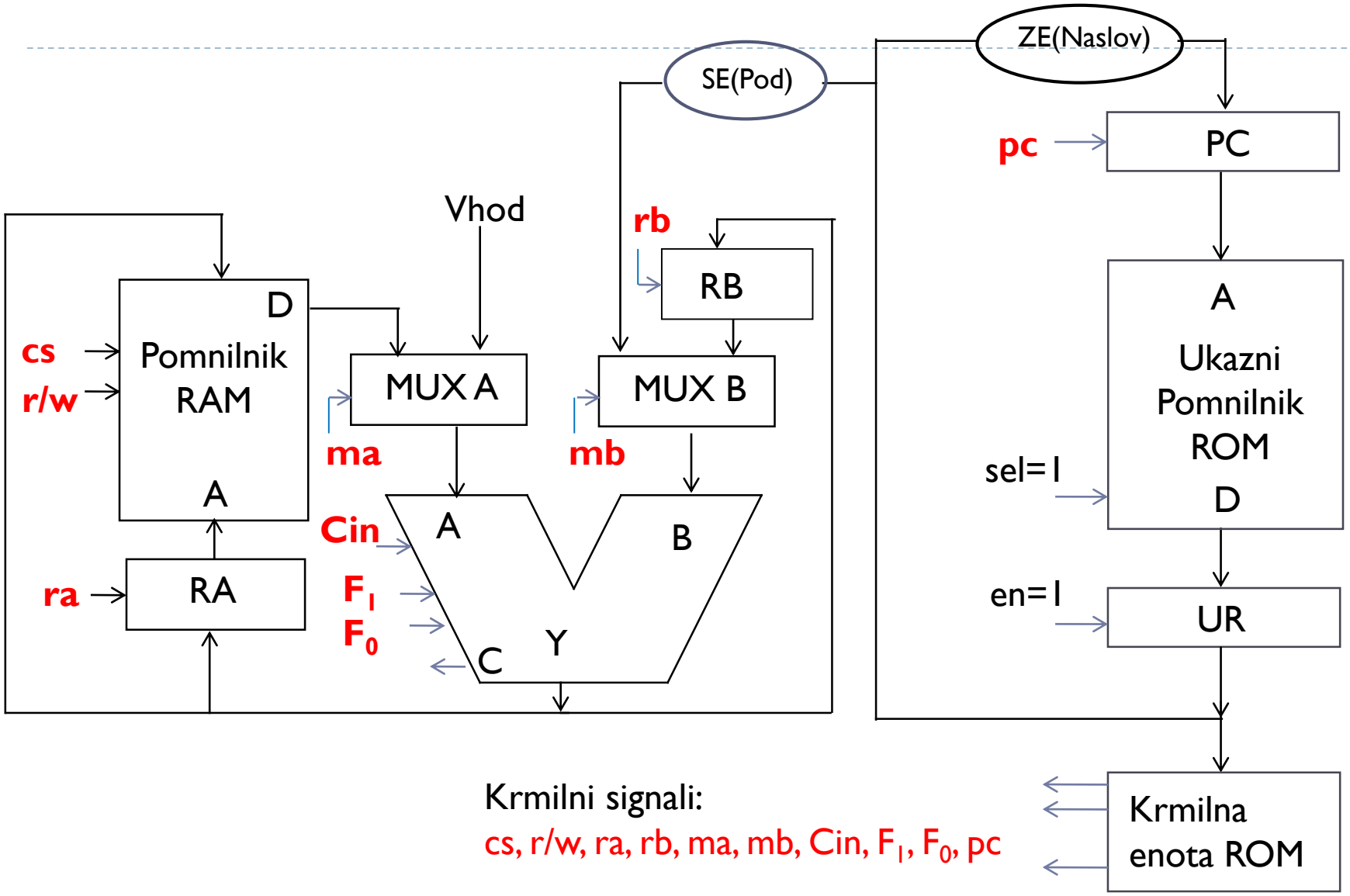
Pomnilnik: **cs** (1 bit)
0: ni omogočen
1: omogočen

Pomnilnik: **r/w** (1 bit)
0: Vpis
1: Branje

Programski števec: **pc** (1 bit)
0: inkrement (+1)
1: vpis (vejitveni naslov)

Vhod števca ct bo določen iz krmilne enote (pc), Load pa bo negacija tega signala.

4- Logična shema procesorja s krmilnimi signali



Krmilni signali:

cs, r/w, ra, rb, ma, mb, Cin, F_1, F_0, pc

5- Krmilni signali za podan nabor ukazov

- Za vsak ukaz so določeni krmilni signali, ki določajo izvajanje.
- Ukaz 0000 je mirovna kombinacija, ki se nastavi ob vklopu.

Ukaz	ra	rb	ma	mb	cin	F_1	F_0	cs	r/w	pc	
	9	8	7	6	5	4	3	2	1	0	ROM_H
0000	0	0	- (0)	- (0)	- (0)	- (0)	- (0)	0	- (0)	0	000_H
18xx	0	1	0	0	0	0	0	1	1	0	106_H
29xx	0	1	1	0	0	0	1	1	1	0	$18e_H$
3bxx	0	1	- (0)	0	1	1	1	1	1	0	$13e_H$
47xx	1	0	- (0)	1	0	1	1	1	1	0	$25e_H$
4bxx	0	1	- (0)	1	0	1	1	1	1	0	$15e_H$
53xx	0	0	- (0)	1	0	1	1	1	0	0	$05c_H$
60xx	0	0	- (0)	- (0)	- (0)	- (0)	- (0)	1	1	1	007_H

Ukaz: OP KODA , R, F, Podatek/Naslov (šestnajstiški zapis)

- v realizaciji se določi 0

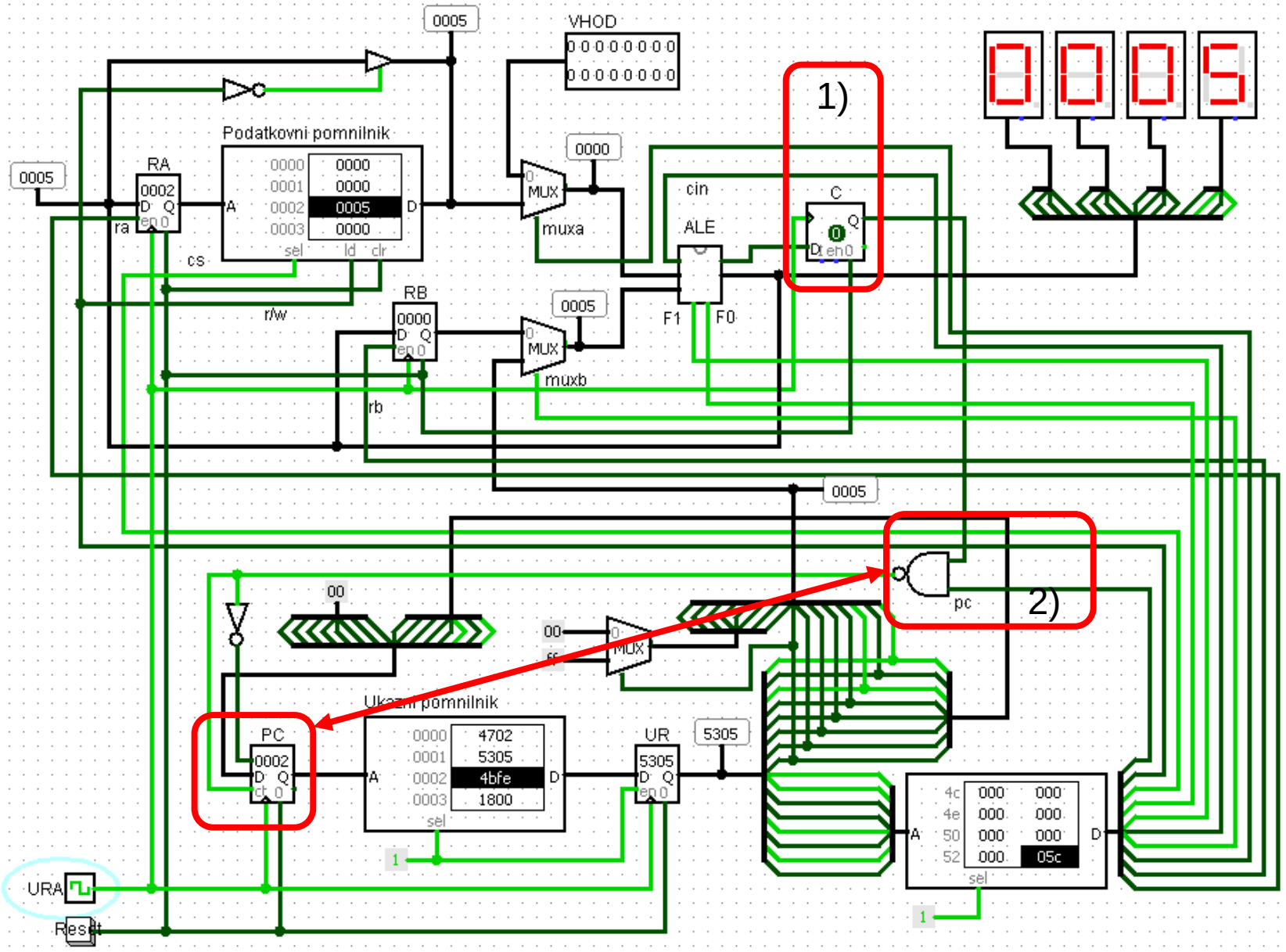
Krmilni signali se vpišejo v ROM (šestnajstiški zapis)

6- Krmilna enota – pomnilnik ROM

- Vpis krmilnih signalov v ROM:
  - zgornjih 8-bitov ukaza (OP KODA, R, F) določa naslov pomnilne besede, kjer se shranijo vrednosti krmilnih signalov.
  - Velikost pomnilnika ROM je 256 x 10

Ukaz Naslov	Ukaz	Krmilni signali (dvojiško)	Krmilni signali (Hex)
00	NOP	00 0000 0000	000
18	ADD RB,Vhod	01 0000 0110	106
29	SUB RB, (RA)	01 1000 1110	18e
3b	INC RB	01 0011 1110	13e
47	MOVE RA, #Pod	10 0101 1110	25e
4b	MOVE RB, #Pod	01 0101 1110	15e
53	MOVE (RA), #Pod	00 0101 1100	05c
60	BRC Naslov	00 0000 0111	007

7- Implementacija v logisimu



Implementirati moramo dodatno vezje za:

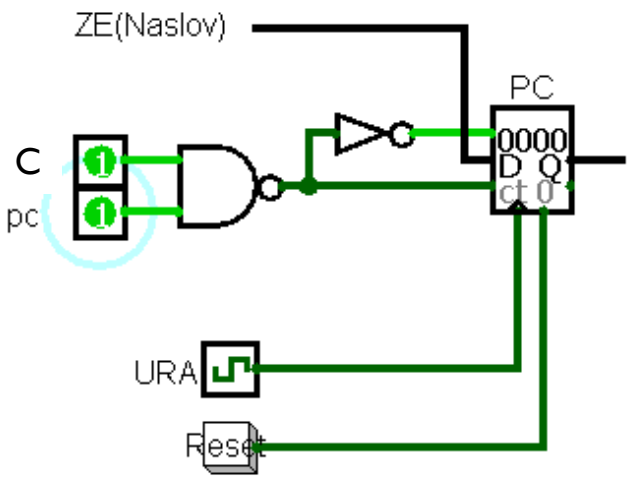
1. Shranjevanje zastavice za prenos (C) v pomnilno celico na izhodu ALE po aritmetični operaciji seštevanja (A+B).
2. Krmiljenje programskega števca (PC) za izvedbo vejitvenega ukaza v odvisnosti od krmilnega signala **pc** (0: inkrement PC, 1: vpis vejitvenega naslova) in od zastavice C (vejitev se izvede, če je C=1).

Definirajmo nov signal, ki določa delovanje PC:

- ct = 1 - inkrement števca PC=PC+1
- ct = 0 - vpis vejitvenega naslova ZE(Naslov)

pc	C	ct
0	0	1
0	1	1
1	0	1
1	1	0

$$ct = \overline{pc.C}$$
$$Load = \overline{ct}$$



8 - Testni program

Naslov (Hex)	Ukaz	Ukaz (Hex)	Opis
0	MOVE RA, 02	4702	V reg RA se vpiše 2
1	MOVE (RA), 05	5305	Na naslov 2 v RAM se vpiše 5
2	MOVE RB, FE	4BFE	V reg RB se vpiše FE_{16}
3	ADD RB, Vhod	1800	Reg RB se prišteje Vhod
4	BRC 10	6010	Skok na naslov 10_{16}, če je zastavica C=1
5	INC RB	3B00	Povečanje reg RB za 1
...	...	...	...
10	SUB RB, (RA)	2900	Od reg RB se odšteje podatek na naslovu v reg RA
11	INC RB	3B00	Povečanje reg RB za 1

- Vpis v registre je določen s signalom URA:
  - PC, RA, RB, C – pozitivna (prednja) fronta (0 v 1)
  - UR – negativna (zadnja) fronta (1 v 0)

Naloga 1

- Delovanje 16-bitnega procesorja - zapis rezultatov v tabeli.
- Start: URA=1, Reset, PC=0, RA=0, RB=0, C=0, RAM:

Naslov (Hex)	Ukaz (Hex)	Ukaz	PC (HEX)	RB (HEX)	RA (Hex)	C-FF (Q)	RAM (Hex)
0	4702	MOVE RA, 02					
1	5305	MOVE(RA), 05					
2	4BFE	MOVE RB, FE					
3	1800	ADD RB,Vhod = 2					
4	6010	BRC 10					
5	3B00	INC RB					
...	...	...					
10	2900	SUB RB, (RA)					
11	3B00	INC RB					

Naloga 1 Rešitev

- Delovanje 16-bitnega procesorja.
- Testiranje obstoječega programa in zapis rezultatov za PC, RB, RA, C-FF ina RAM (za naslov v RA) po izvedbi ukaza v tabeli.
- Start programa: URA=1, Reset, PC=0, RA=0, RB=0, C=0, RAM = 00..:

Naslov (Hex)	Ukaz (Hex)	Ukaz	PC (HEX)	RB (HEX)	RA (Hex)	C-FF (Q)	RAM (Hex)
0	4702	MOVE RA, 02	1	0	2	0	0
1	5305	MOVE(RA), 05	2	0	2	0	5
2	4BFE	MOVE RB, FE	3	FFFE	2	0	5
3	1800	ADD RB,Vhod = 2	4	0000	2	1	5
4	6010	BRC 10	10	0000	2	0	5
5	3B00	INC RB	6	0000	2	0	5
...	...	...					
10	2900	SUB RB, (RA)	11	FFFB	2	0	5
11	3B00	INC RB	12	FFFC	2	0	5

Naloga 2

- Nadgradimo procesor s spodnjimi 3 ukazi:

OP KODA	Ukaz (mnemonik)	Tip	Funkcija
0111	AND RB, Vhod	D-tip	RB ← RB & Vhod
1000	MOVE RA, RB	P-tip	RA ← RB
1001	BR Naslov	P-tip	PC ← ZE(Naslov)

- Naloge:
  1. Zapis ukazov v tabelo, testni program.
  2. Določitev gradnikov in podatkovnih poti.
  3. Določitev krmilnih signalov in vpis v ROM.
  4. Vpis testnega programa v ROM.
  5. Morebitna dopolnitev vezja zaradi dodanih ukazov.
  6. Zapis rezultatov po vsakem koraku izvajanja.

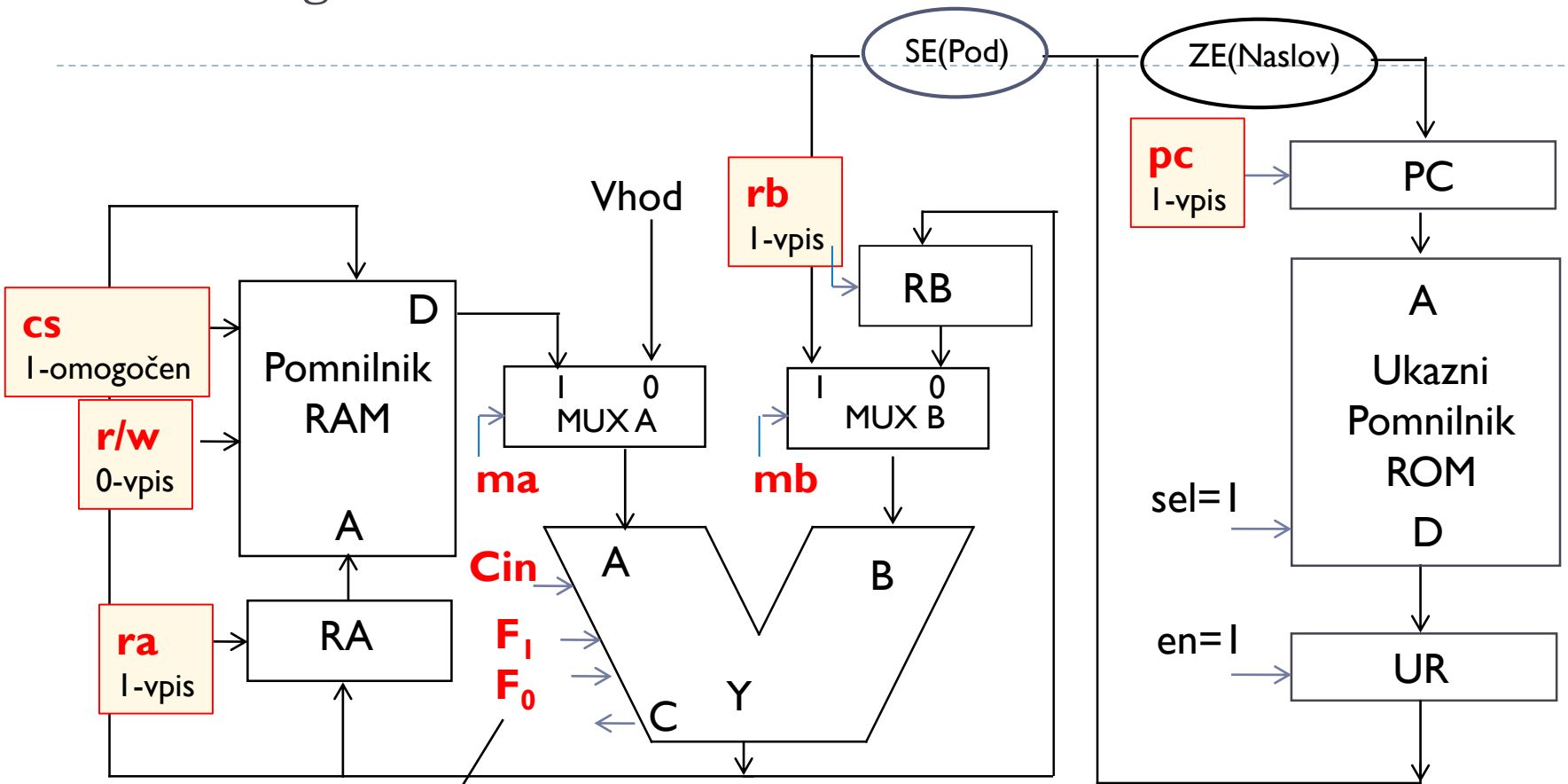
- Zapis ukazov: polja, ki so uporabljena v procesorju.

Ukaz (mnemonik)	OP KODA	Reg	ALE (F)	Cin	Zapis ukaza (b_{15}-b_8)	Zapis ukaza (Hex)
AND RB, Vhod	0111	RB(10)	A & B (10)	-	0111 1010	7axx
MOVE RA, RB	1000	RA(01)	B (11)	0	1000 0111	87xx
BR Naslov	1001	-	-	-	1001 0000	90xx

- Testni program

Naslov	Ukaz (mnemonik)	Ukaz (Hex)	Opis ukaza
0	MOVE RB, #32	4B32	Vpis podatka 32 v register RB
1	AND RB, Vhod	7A00	V register RB se vpiše konjunkcija registra RB in podatka Vhod (0077)
2	MOVE RA, RB	8700	V register RA se vpiše vsebina registra RB
3	BR 0	9000	Skok na naslov 0

Krmilni signali za nove ukaze



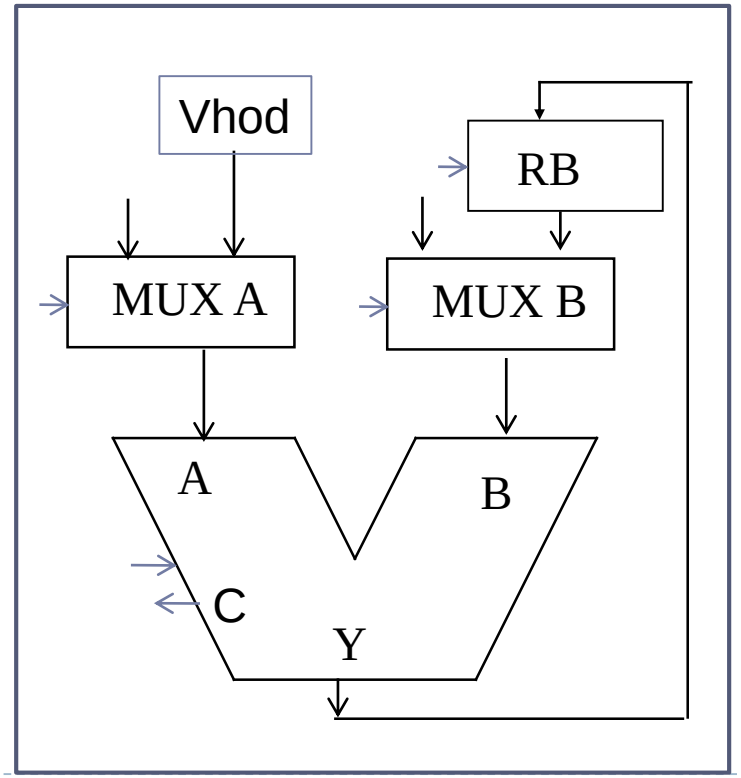
F_1	F_0	Y
0	0	A+B
0	1	B - A
1	0	A & B
1	1	B

Krmilni signali:

cs, r/w, ra, rb, ma, mb, Cin, F_1, F_0, pc

AND RB, Vhod (Podatkovne poti)

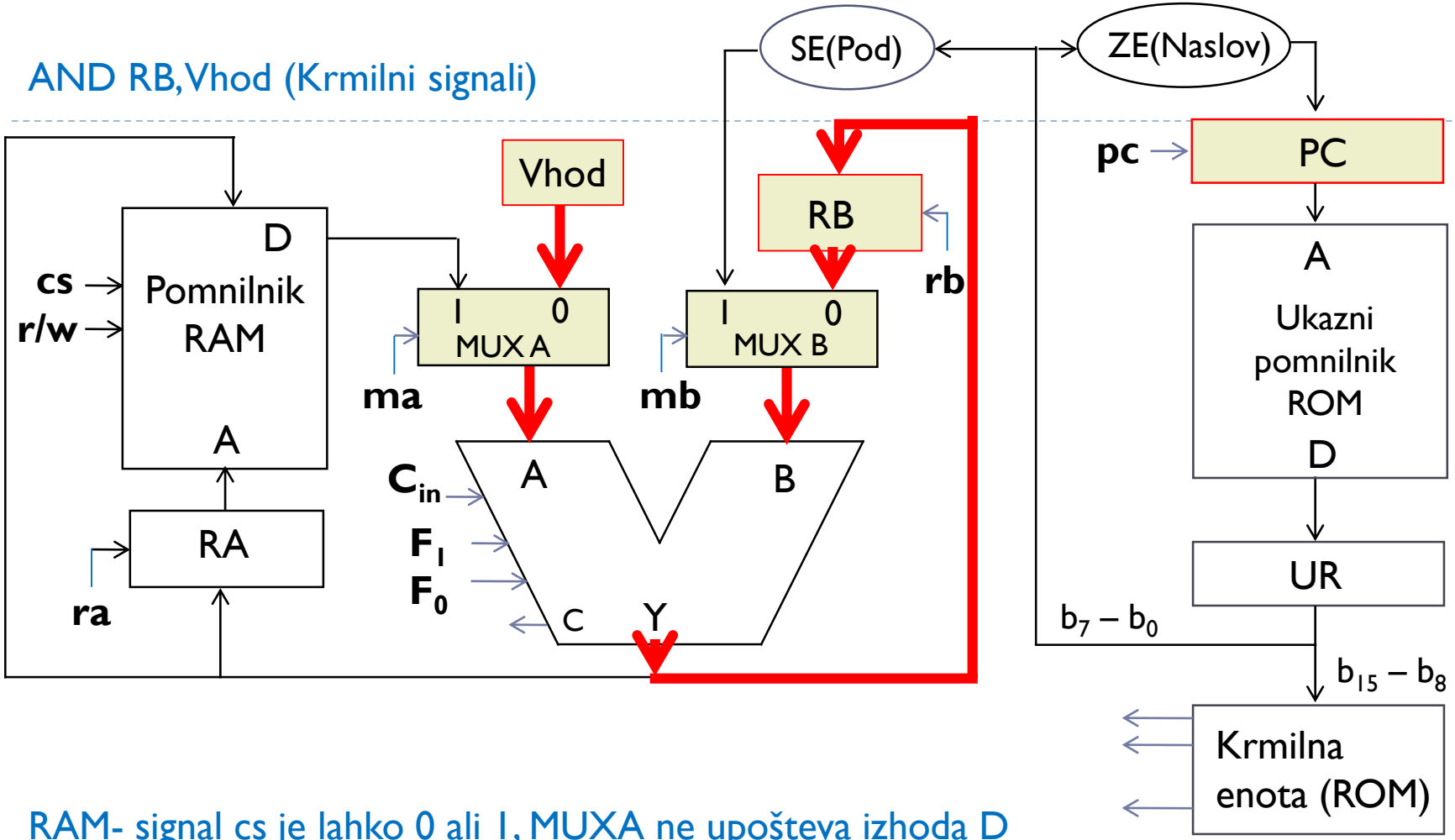
- Na vhodu MUXA bo izbran podatek Vhod (vhod A na ALE).
- Vsebina registra (RB) bo izbrana na vhodu MUXB (vhod B na ALE).
- Izvedba operacije konjunkcije A&B, rezultat bo na izhodu ALE.
- Podatek se shrani v register RB.



AND RB, Vhod

A <= Vhod
B <= RB
Y <= A & B
RB <= Y

AND RB,Vhod (Krmilni signali)

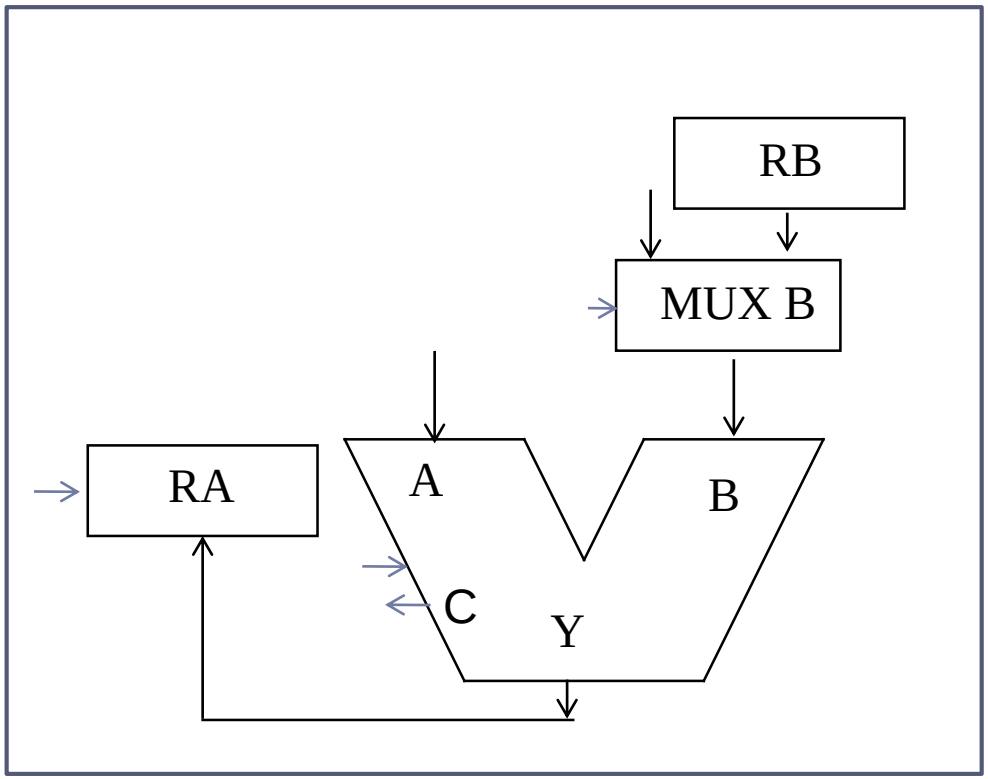


RAM- signal cs je lahko 0 ali 1, MUXA ne upošteva izhoda D

Ukaz	ra	rb	ma	mb	cin	F_1	F_0	cs	r/w	pc	ROM
7a	0	1	0	0	- (0)	1	0	0	1	0	112
7a	0	1	0	0	- (0)	1	0	1	1	0	116

MOVE RA, RB (Podatkovne poti)

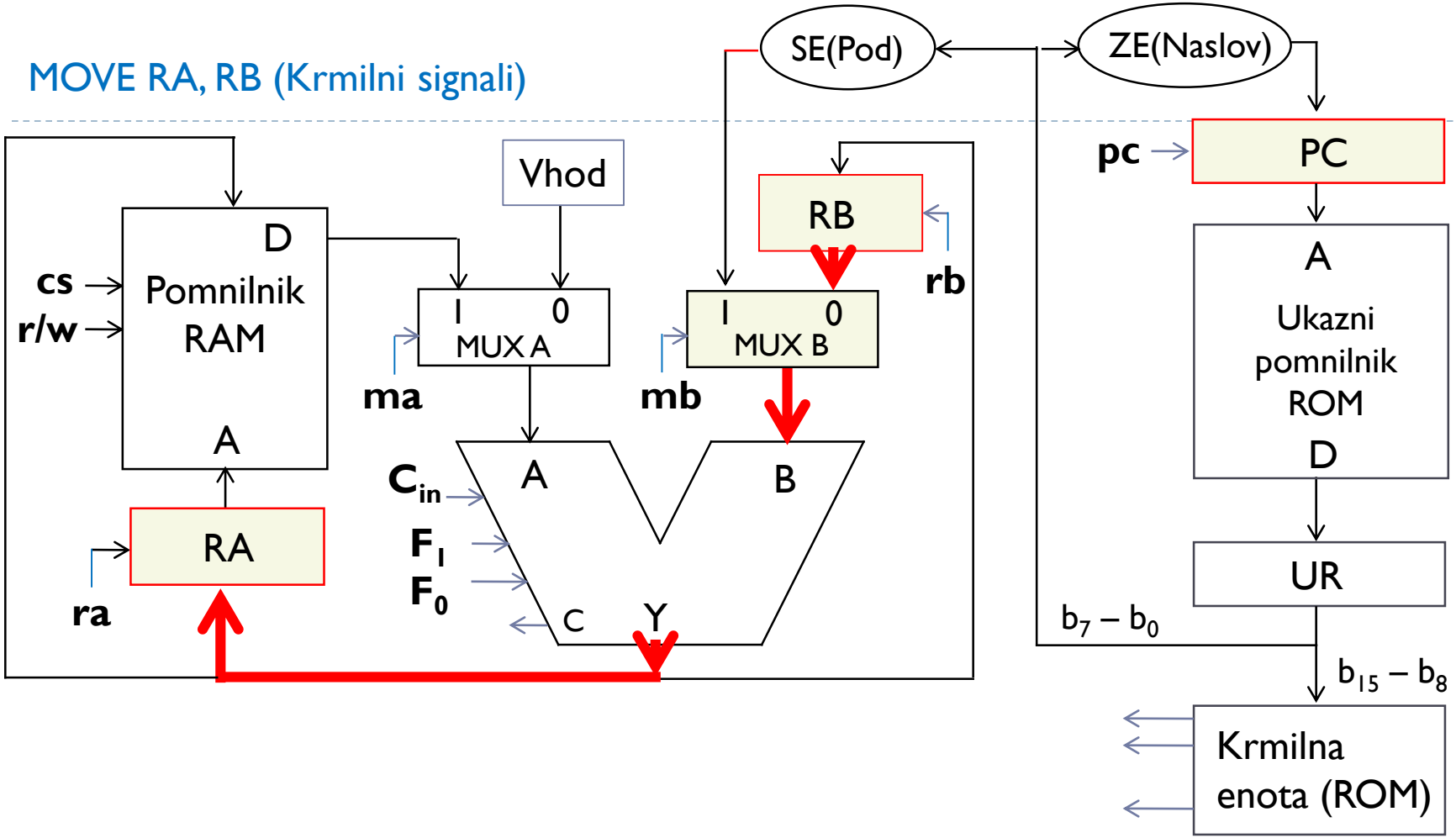
- Vsebina registra (RB) bo izbrana na vhodu MUXB (vhod B na ALE).
- Izvedba operacije B, rezultat bo na izhodu ALE.
- Podatek se shrani v register RA.



MOVE RA, RB

B <= RB
Y <= B
RA <= Y

MOVE RA, RB (Krmilni signali)



Ukaz	ra	rb	ma	mb	cin	F_1	F_0	cs	r/w	pc	ROM
87	1	0	- (0)	0	0	1	1	1	1	0	21e