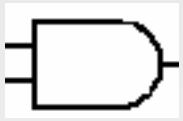
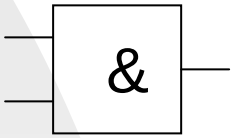
2. Arhitektura mikroprocesorjev

- Osnove digitalnih vezij
- Model mikroprocesorja
  - krmilna enota
  - aritmetično/logična enota (ALE)
  - registri
- Delovanje mikroprocesorja
  - Faze delovanja
  - Oblika strojnih ukazov
  - Implementacija krmilne enote

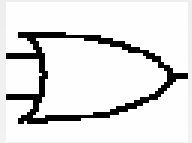
Osnove digitalnih vezij

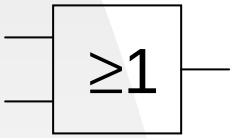
- Logična vrata



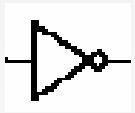


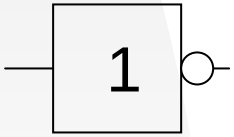
IN (AND)



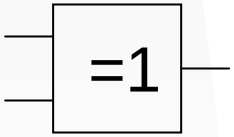


ALI (OR)





NE (NOT)



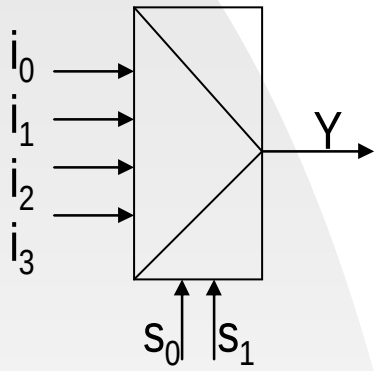
IZKLJUČNI ALI (EXCLUSIVE OR)

A B	A∨B	A∧B	A⊕B	$\bar{A}$
0 0	0	0	0	1
0 1	1	0	1	1
1 0	1	0	1	0
1 1	1	1	0	0

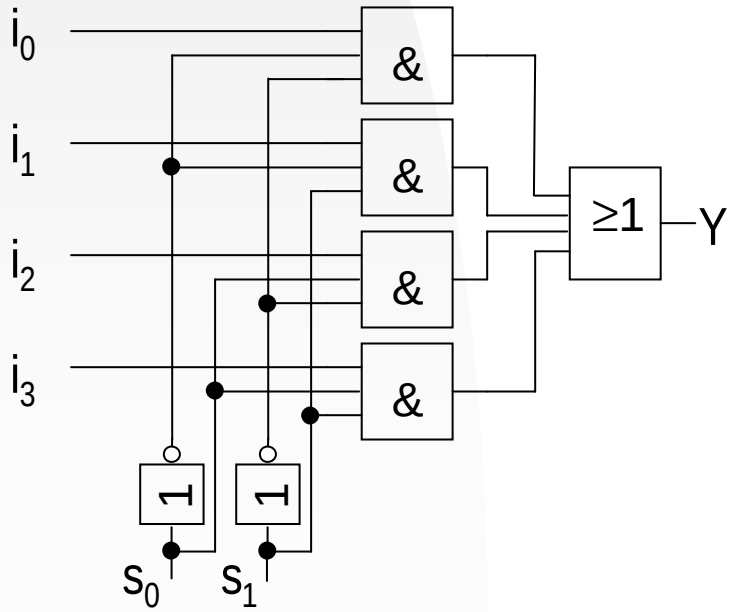
Osnove digitalnih vezij

- Kombinacijska vezja

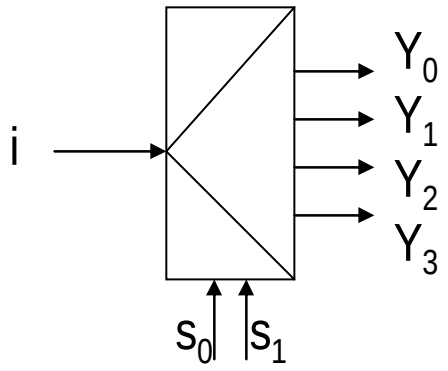
Multiplekser



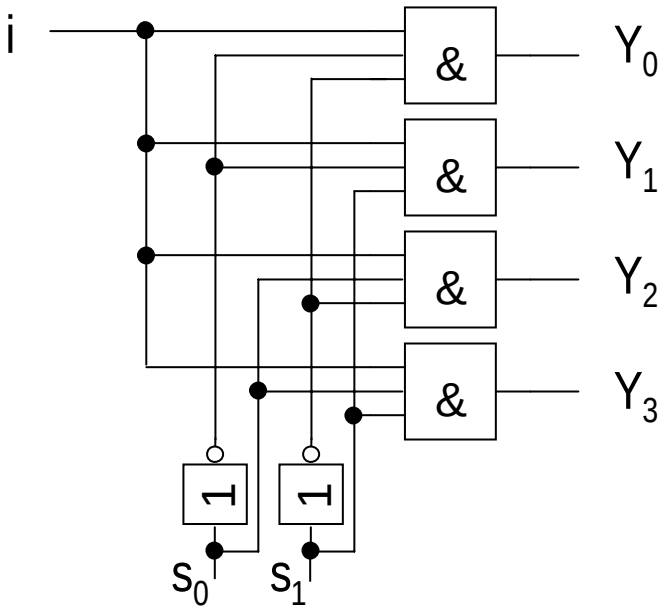
s_0	s_1	Y
0	0	i_0
0	1	i_1
1	0	i_2
1	1	i_3



Demultiplekser



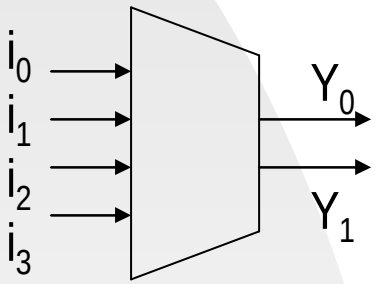
s_0	s_1	Y_0	Y_1	Y_2	Y_3
0	0	i	0	0	0
0	1	0	i	0	0
1	0	0	0	i	0
1	1	0	0	0	i



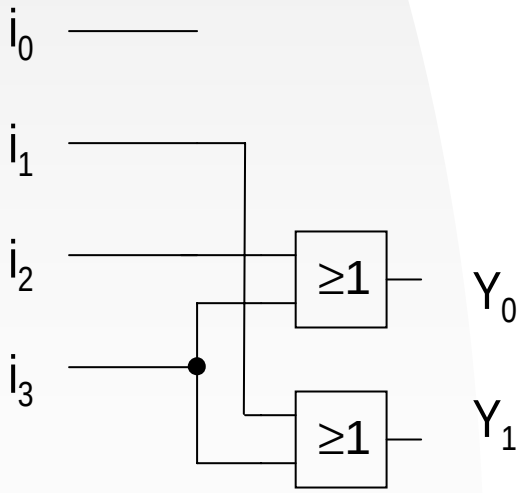
Osnove digitalnih vezij

- Kombinacijska vezja

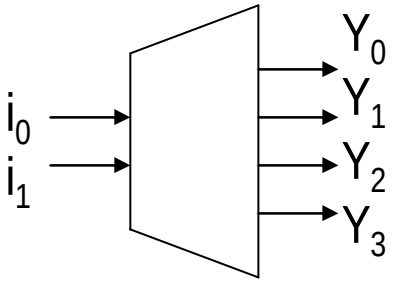
Kodirnik



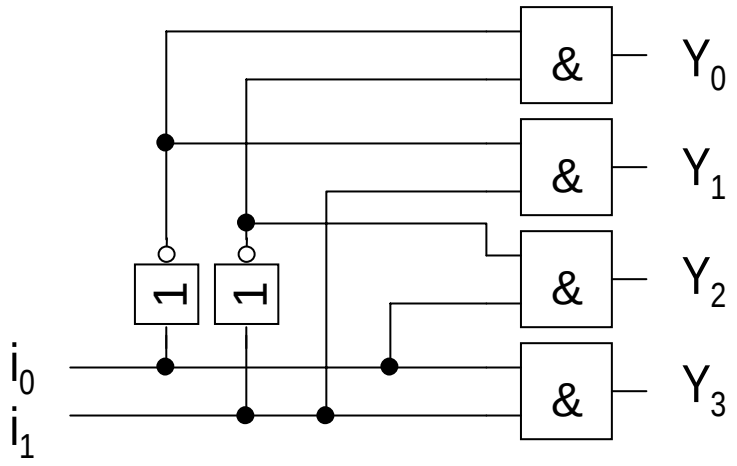
i_0	i_1	i_2	i_3	Y_0	Y_1
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1



Dekodirnik



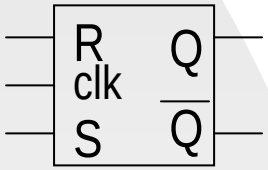
i_0	i_1	Y_0	Y_1	Y_2	Y_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

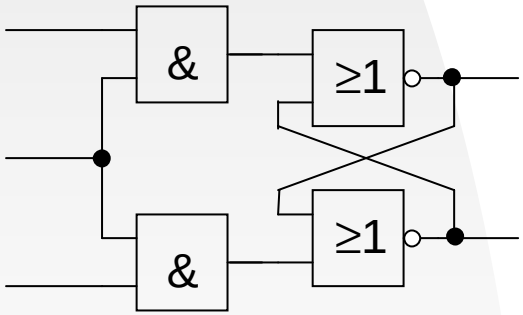


Osnove digitalnih vezij

- Sekvenčna vezja

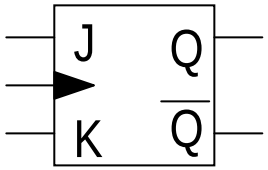
RS flip-flop

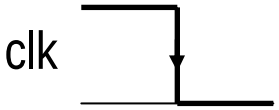




R	S	Q	$\overline{Q}$
0	0	Q	$\overline{Q}$
0	1	0	1
1	0	1	0
1	1	prep.	

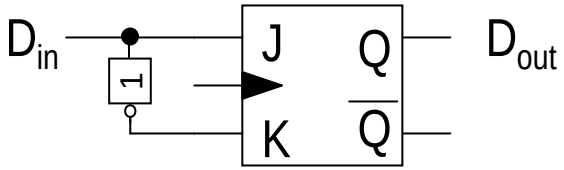
JK flip-flop

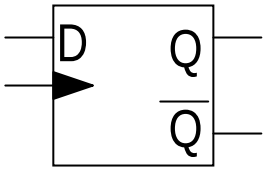




R	S	Q	$\overline{Q}$
0	0	Q	$\overline{Q}$
0	1	0	1
1	0	1	0
1	1	$\overline{Q}$	Q

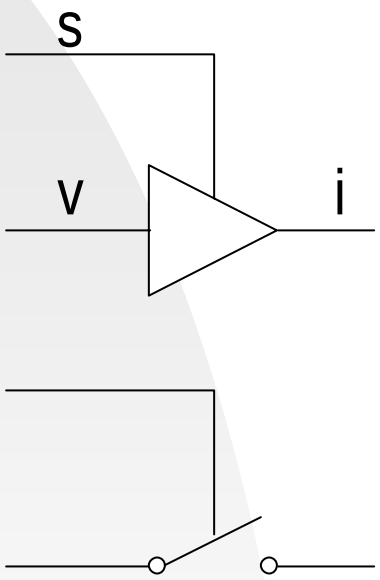
D flip-flop





Osnove digitalnih vezij

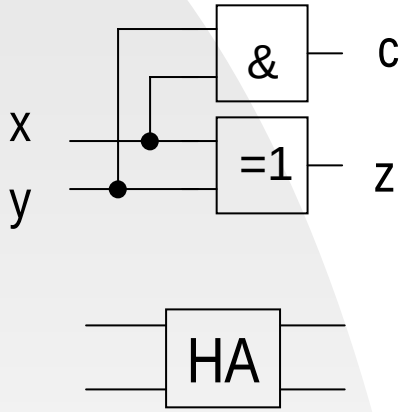
- Stikalo



Osnove digitalnih vezij

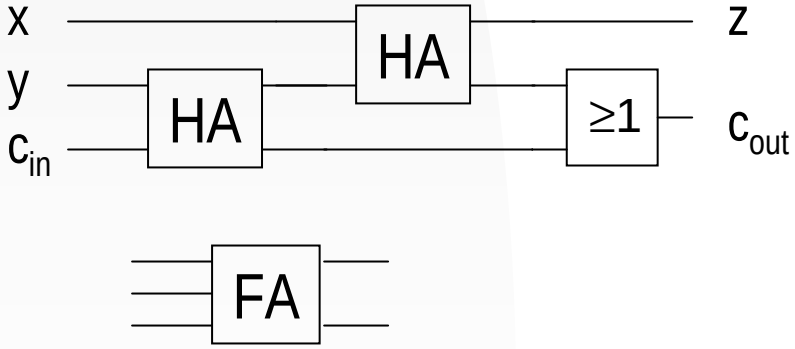
- Zanimivejše kombinacije

Polovični seštevalnik

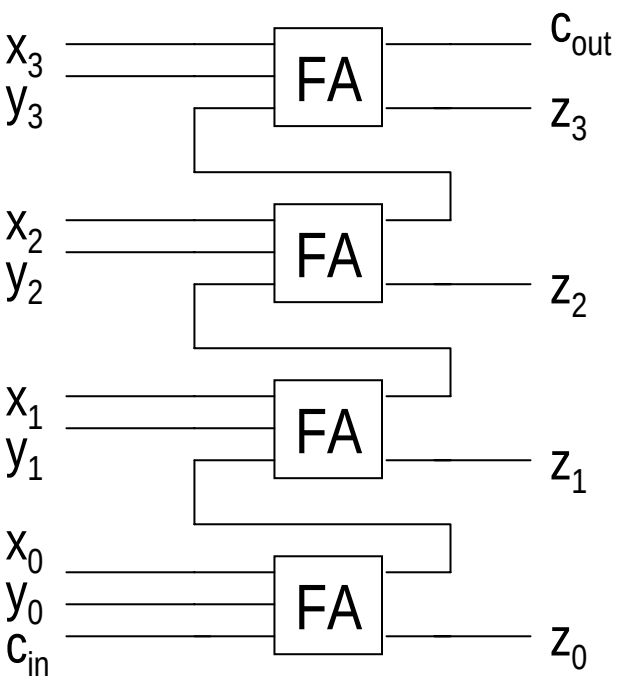


x y	z c
0 0	0 0
0 1	1 0
1 0	1 0
1 1	0 1

Celi seštevalnik



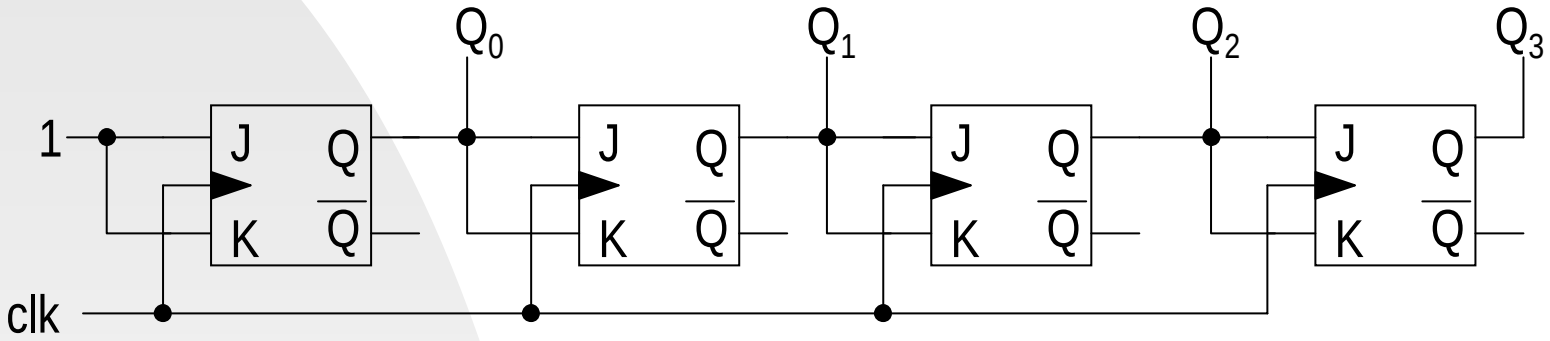
x y c_{in}	z c_{out}
0 0 0	0 0
0 0 1	1 0
0 1 0	1 0
0 1 1	0 1
1 0 0	1 0
1 0 1	0 1
1 1 0	0 1
1 1 1	1 1



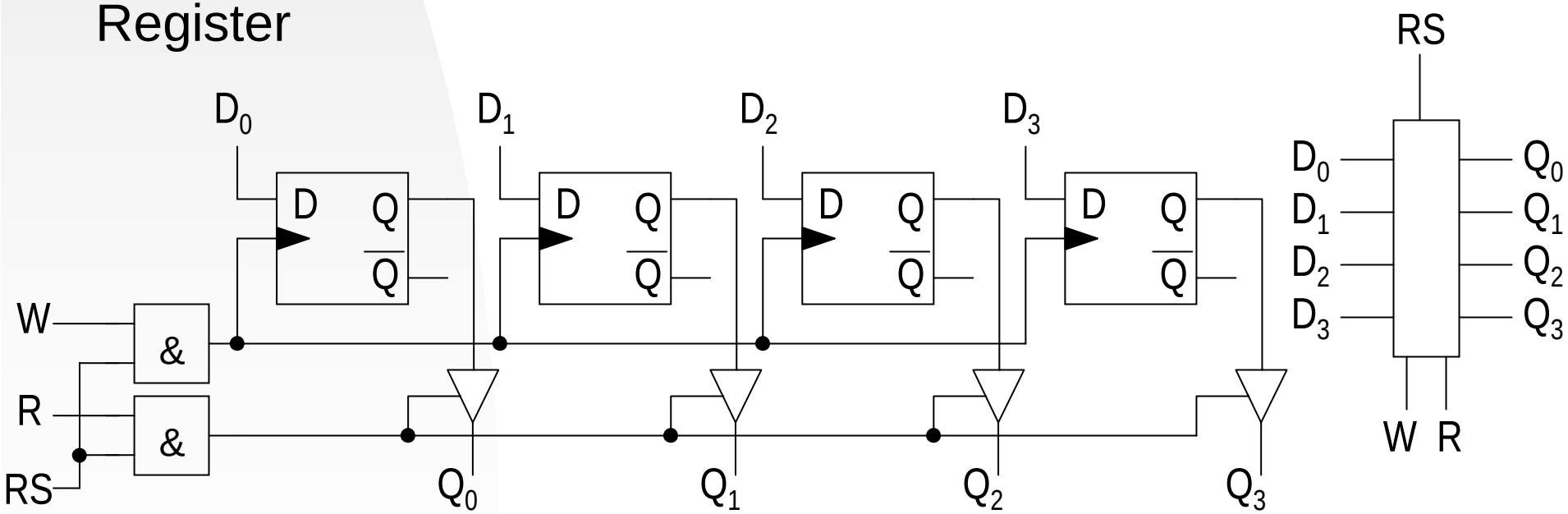
Osnove digitalnih vezij

- Zanimivejše kombinacije

Števec



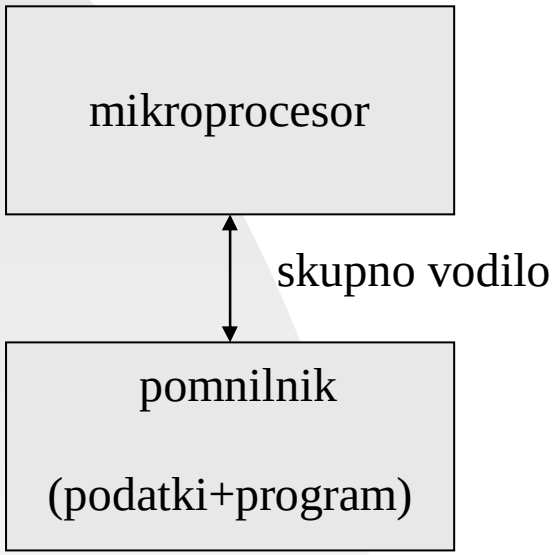
Register



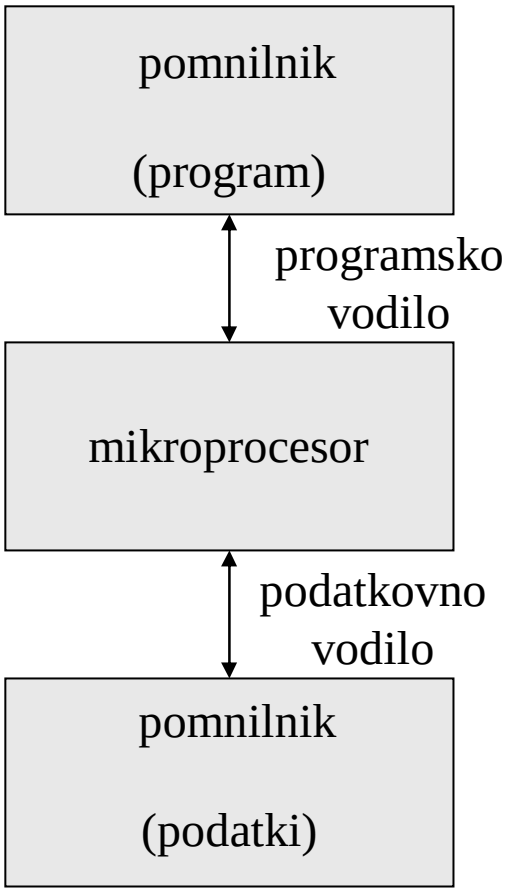
Model mikroprocesorja

- Arhitekture mikroprocesorjev se zelo razlikujejo
  - von Neumannov model: podatki in koda v istem pomnilniku (Pentium)
  - Harwardski model: podatki in koda v ločenih pomnilnikih (PIC)

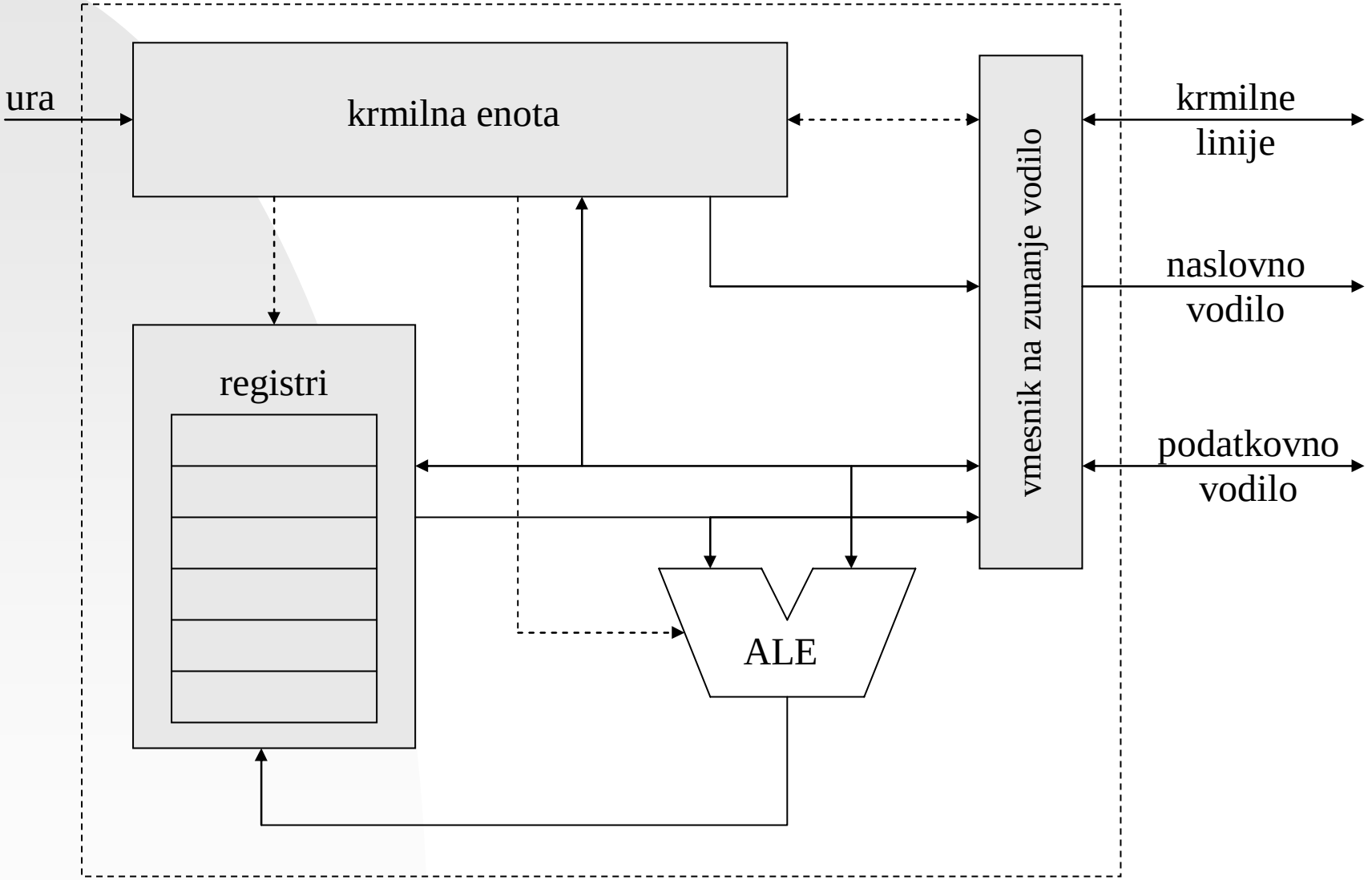
von Neumannov model

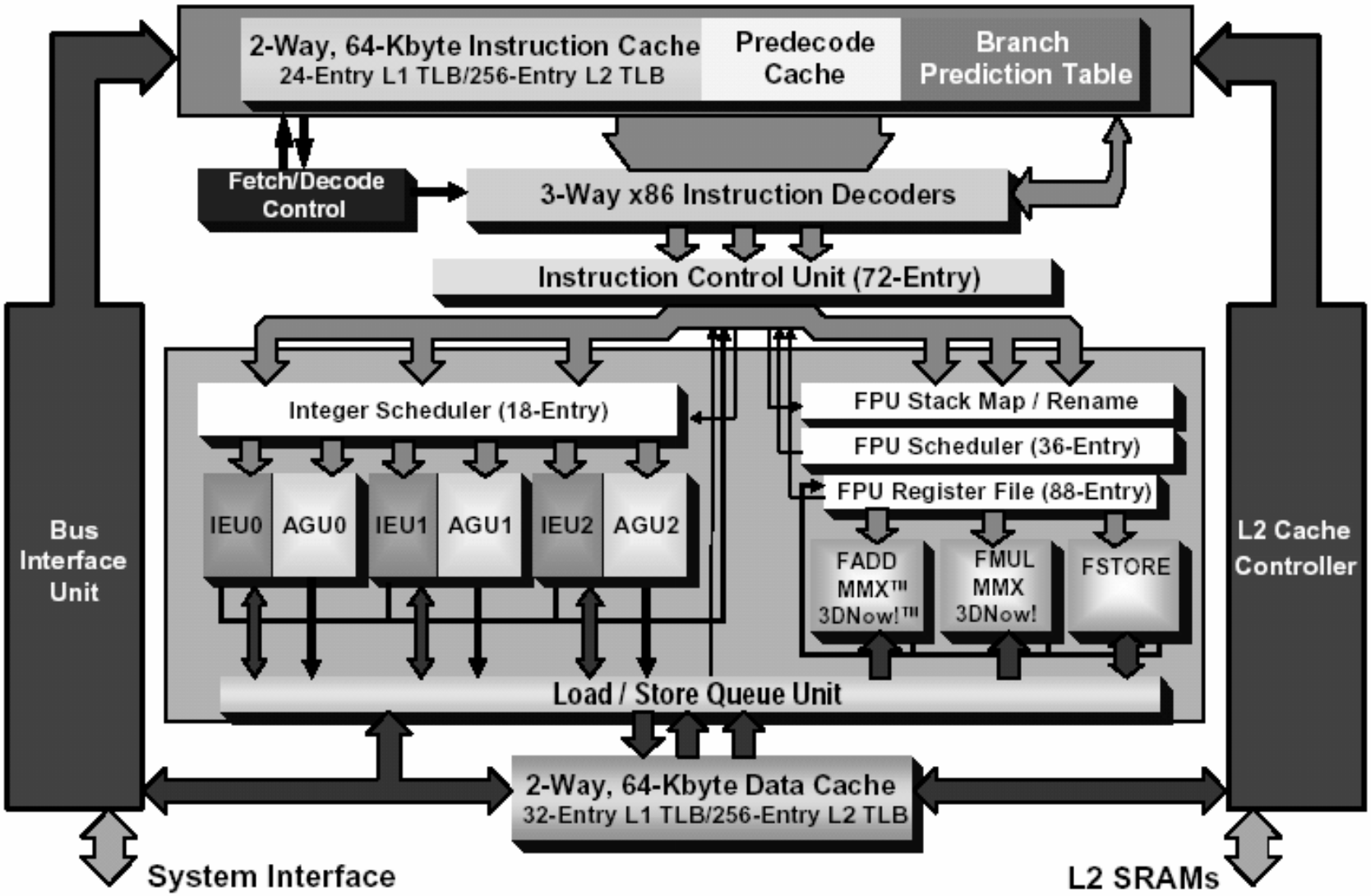


Harwardski model



Hipotetični model mikroprocesorja

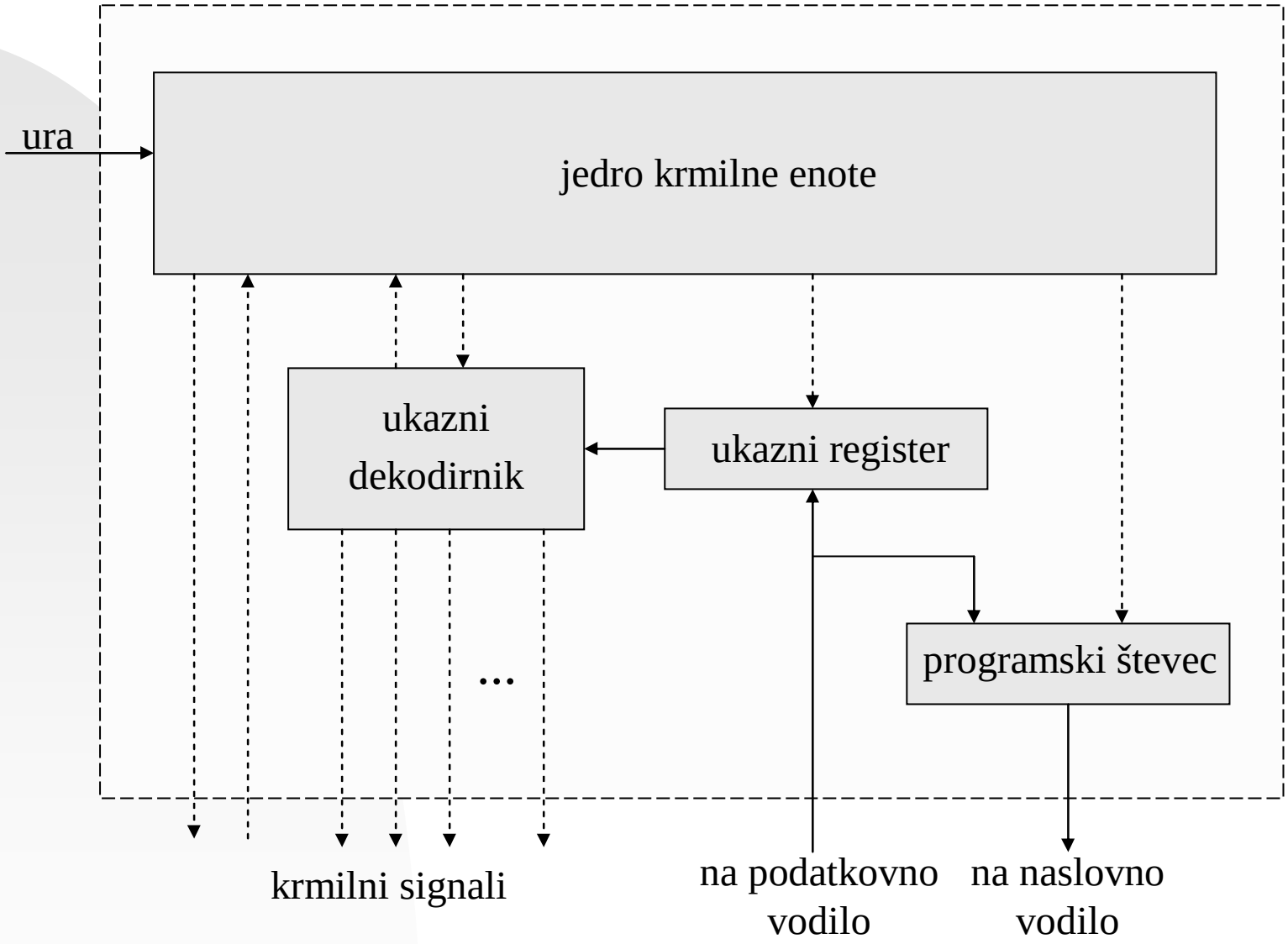



2-Way, 64-Kbyte Instruction Cache
24-Entry L1 TLB/256-Entry L2 TLB
Predecode Cache
Branch Prediction Table
Fetch/Decode Control
3-Way x86 Instruction Decoders
Instruction Control Unit (72-Entry)
Integer Scheduler (18-Entry)
IEU0
AGU0
IEU1
AGU1
IEU2
AGU2
FPU Stack Map / Rename
FPU Scheduler (36-Entry)
FPU Register File (88-Entry)
FADD MMX™ 3DNow!™
FMUL MMX 3DNow!
FSTORE
Load / Store Queue Unit
Bus Interface Unit
L2 Cache Controller
2-Way, 64-Kbyte Data Cache
32-Entry L1 TLB/256-Entry L2 TLB
System Interface
L2 SRAMs

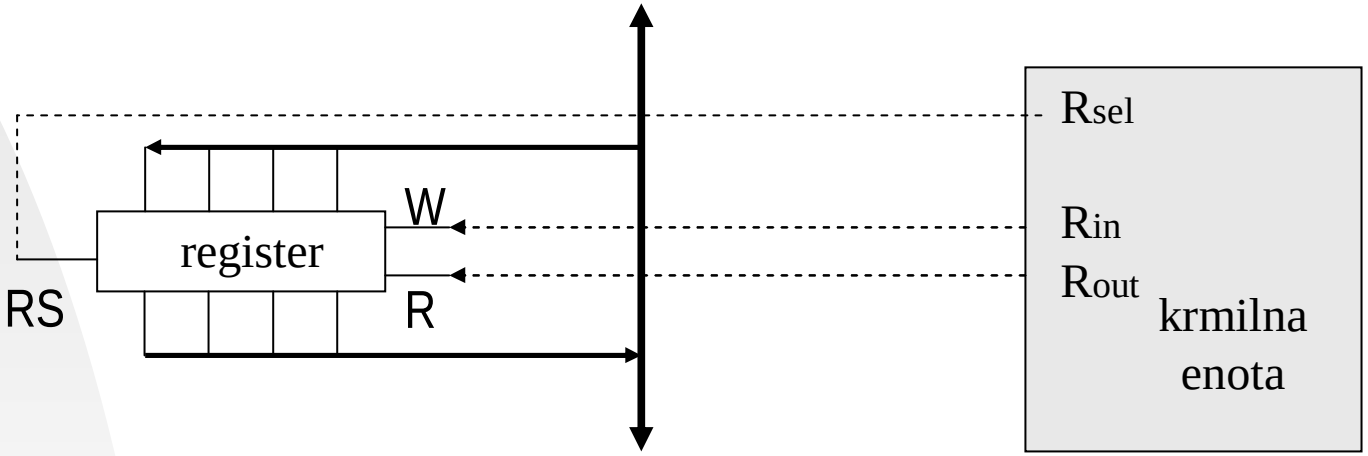
AMD Athlon™

Krmilna enota

- Upravlja delovanje mikroprocesorja
- Sinhrono vezje, ki deluje po taktu ure
- Sestavljena iz:
  - jedra krmilne enote: krmili delovanje mikroprocesorja
  - ukaznega registra: sprejme naslednji ukaz, ki ga mora mikroprocesor izvesti
  - ukaznega dekodirnika: dekodira ukaz in generira krmilne signale za njegovo izvedbo
  - programskega števca: določa naslov naslednjga ukaza, ki se mora izvesti

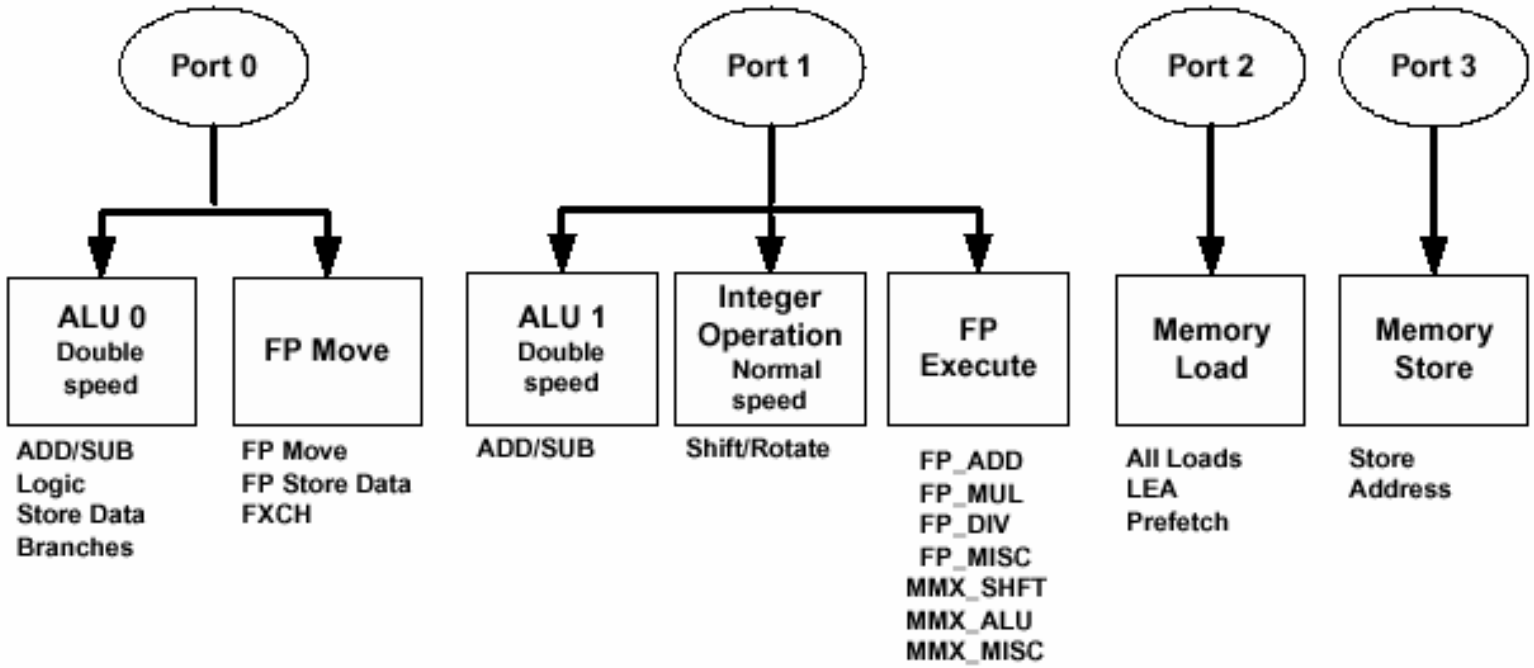

ura
jedro krmilne enote
ukazni
dekodirnik
ukazni register
programski števec
...
krmilni signali
na podatkovno
vodilo
na naslovno
vodilo

Prikaz delovanja krmilnih signalov



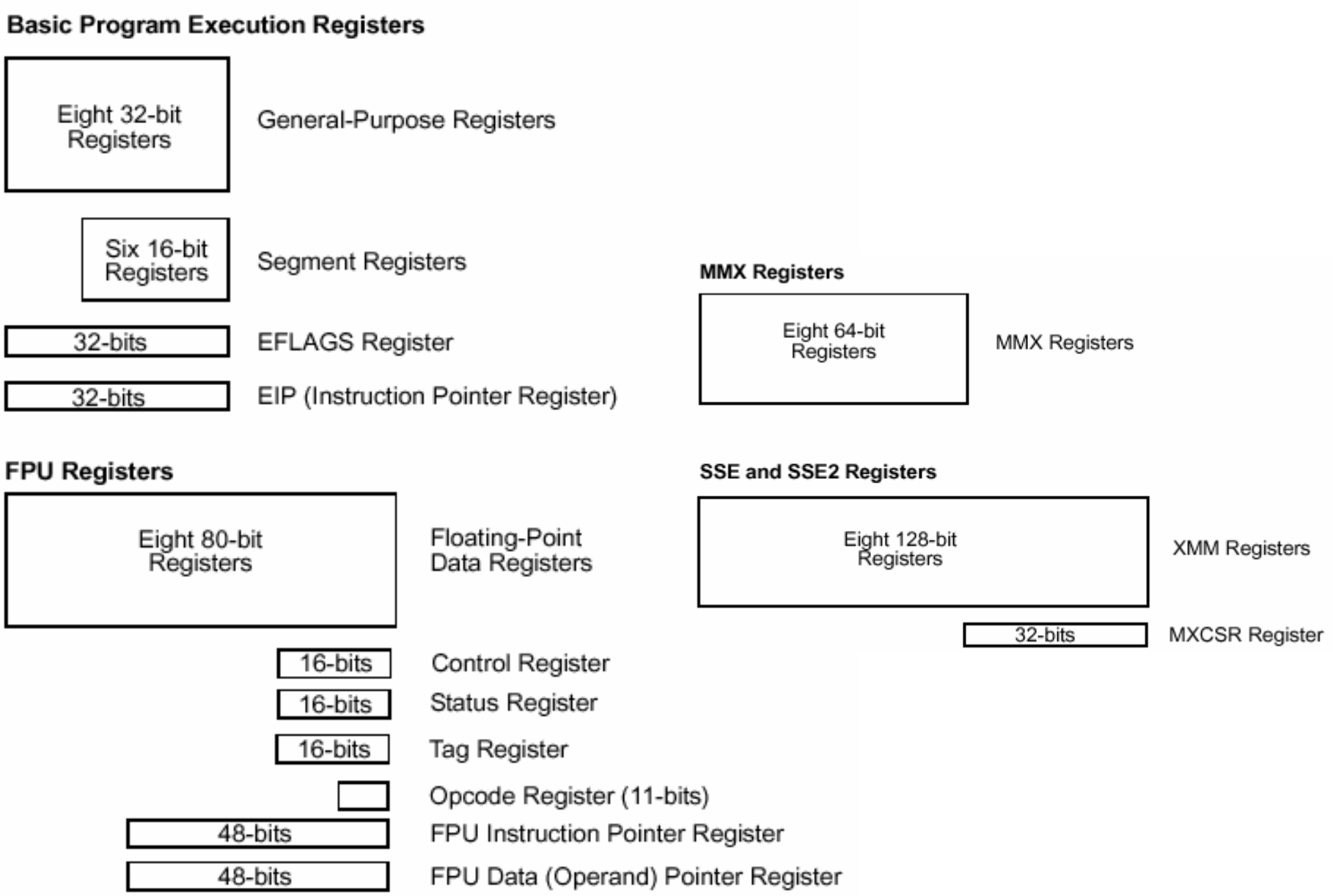
Aritmetično/logična enota (ALE)

- Izvajanje operacij nad operandi mikroprocesorja (matematične in logične operacije)
- Sodobnejši mikroprocesorji imajo več ALE (za cela in realna števila) in podpirajo kompleksnejše operacije (sin, log, …)
- Izvedena kot čisto preklopno vezje


Port 0
Port 1
Port 2
Port 3
ALU 0
Double speed
FP Move
ALU 1
Double speed
Integer Operation
Normal speed
FP Execute
Memory Load
Memory Store
ADD/SUB
Logic
Store Data
Branches
FP Move
FP Store Data
FXCH
ADD/SUB
Shift/Rotate
FP_ADD
FP_MUL
FP_DIV
FP_MISC
MMX_SHFT
MMX_ALU
MMX_MISC
All Loads
LEA
Prefetch
Store
Address

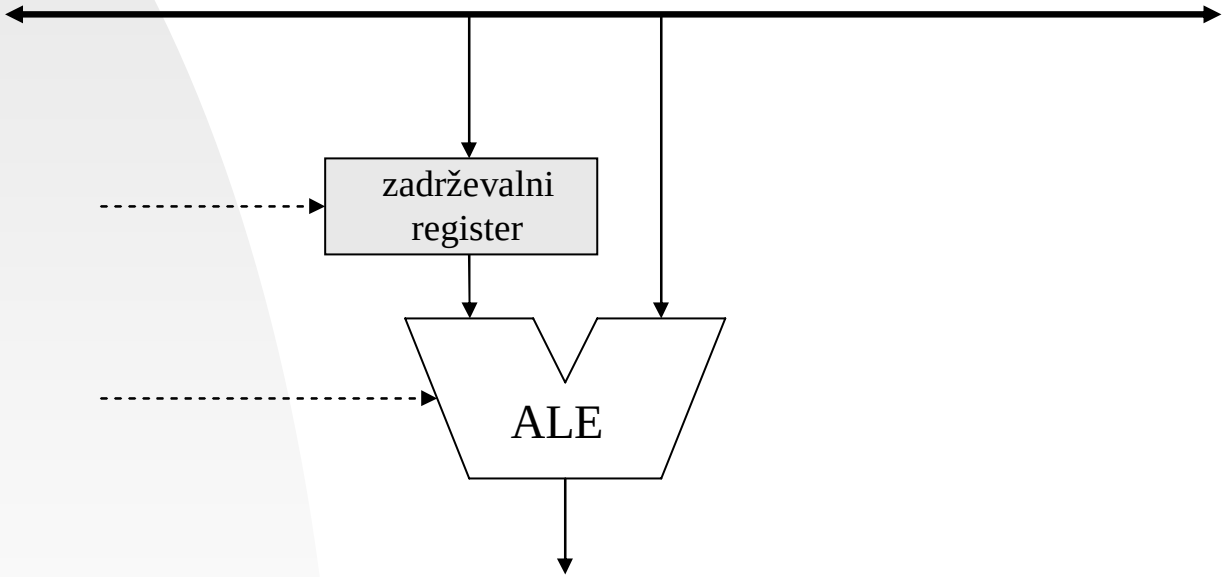
Registri

- Hitre pomnilniške celice znotraj mikroprocesorja
- Osnovne skupine:
  - Podatkovni registri: operandi pri ukazih, shramba
  - Naslovni registri: določajo operande v pomnilniku
  - Posebni registri: PC, UR, ...


Basic Program Execution Registers
Eight 32-bit Registers
General-Purpose Registers
Six 16-bit Registers
Segment Registers
32-bits
EFLAGS Register
32-bits
EIP (Instruction Pointer Register)
MMX Registers
Eight 64-bit Registers
MMX Registers
FPU Registers
Eight 80-bit Registers
Floating-Point Data Registers
16-bits
Control Register
16-bits
Status Register
16-bits
Tag Register
Opcode Register (11-bits)
48-bits
FPU Instruction Pointer Register
48-bits
FPU Data (Operand) Pointer Register
SSE and SSE2 Registers
Eight 128-bit Registers
XMM Registers
32-bits
MXCSR Register

Prenos podatkov znotraj mikroprocesorja

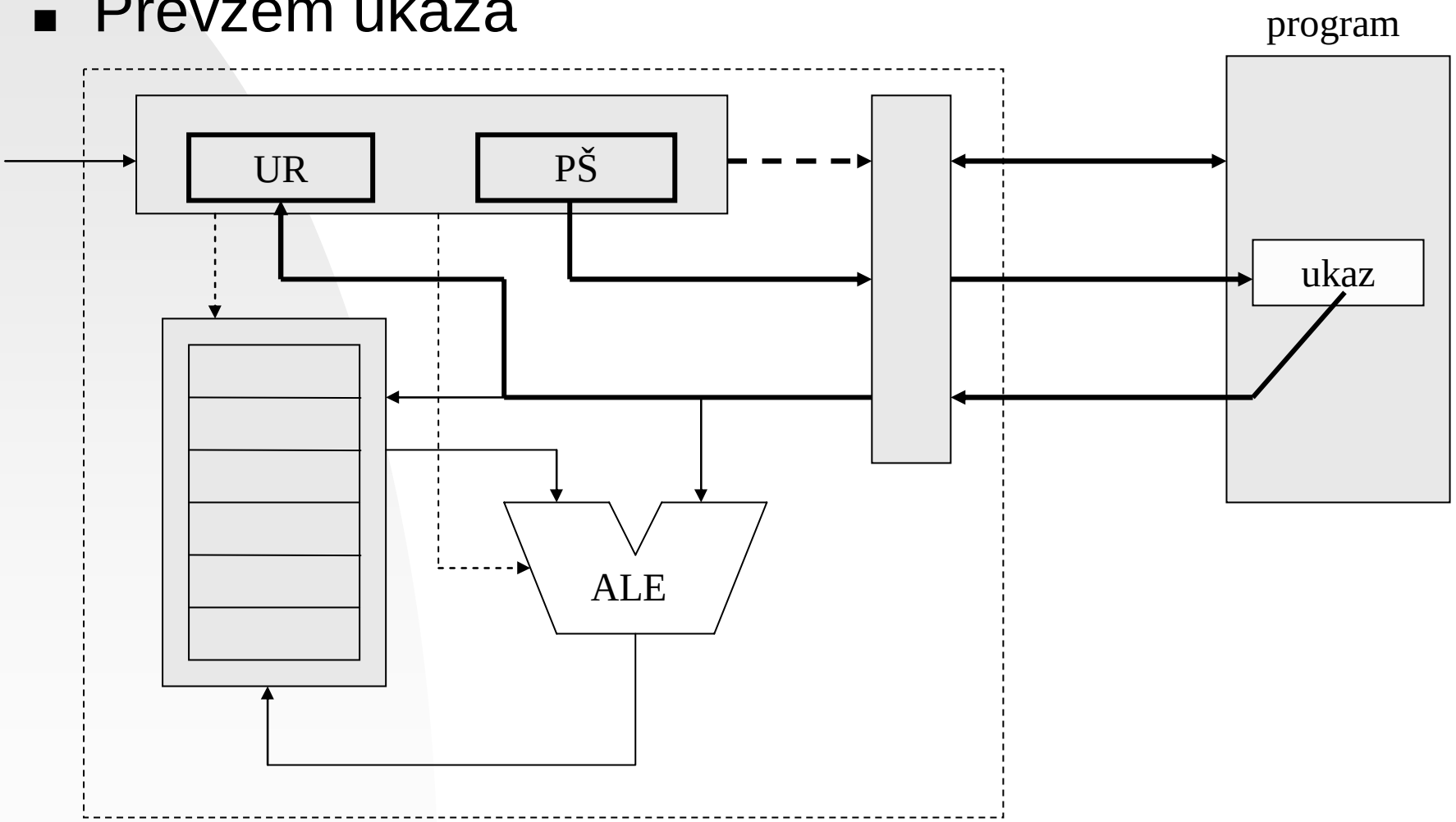
- Interna vodila
  - skupno notranje vodilo



  - ločena notranja vodila

Faze delovanja mikroprocesorja

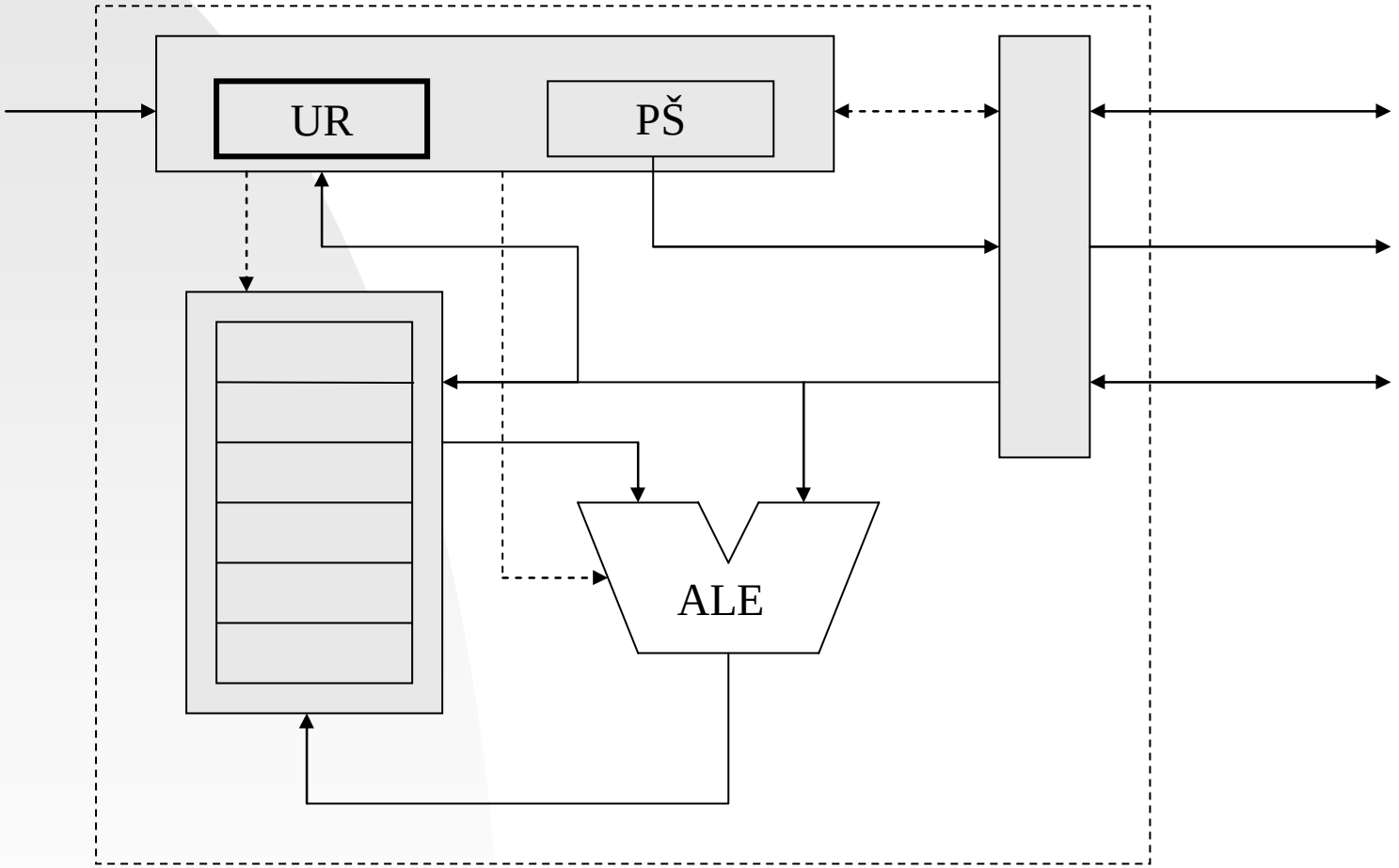
- Prevzem ukaza



Prevzem ukaza

- Na naslovno vodilo se prenese vsebina programskega števca (naslovi se ena izmed celic programskega pomnilnika)
- Krmilne linije se postavijo tako, da se izvede čitanje iz naslovljene lokacije (prečita se koda ukaza, ki ga mora mikroprocesor izvesti)
- Vsebina prečitane lokacije se zapiše v ukazni register

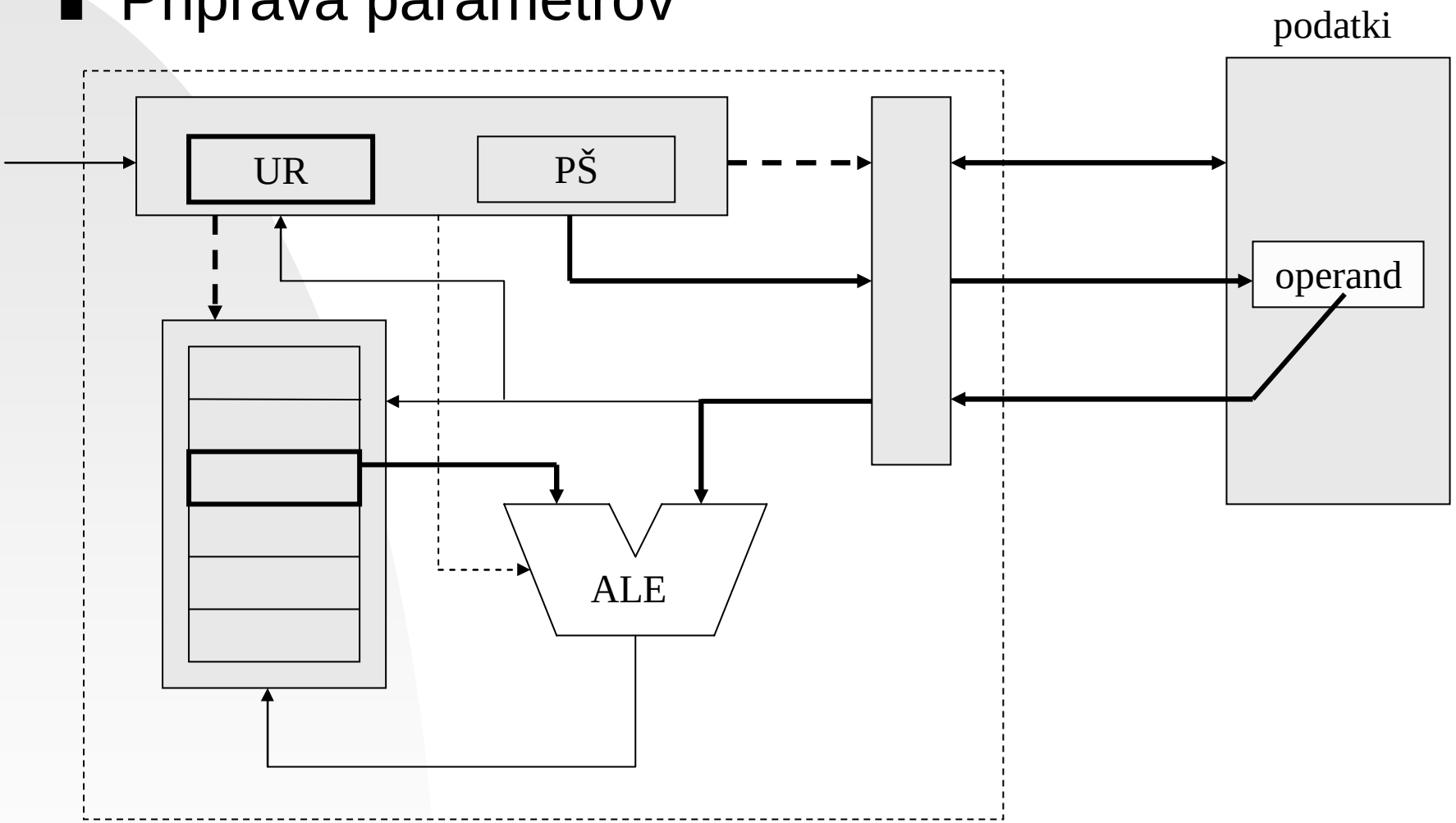
- Dekodiranje ukaza



Dekodiranje ukaza

- Vzorec bitov, ki je zapisan v ukaznem registru se prevede na zaporedje krmilnih signalov
- Bitni vzorec določa katere enote bodo sodelovale v izvedbi ukaza in v kakšnem vrstnem redu
- Pretvorba kode ukaza v zaporedje signalov je odvisna od izvedbe krmilne enote (ožičena logika ali mikroprogram)

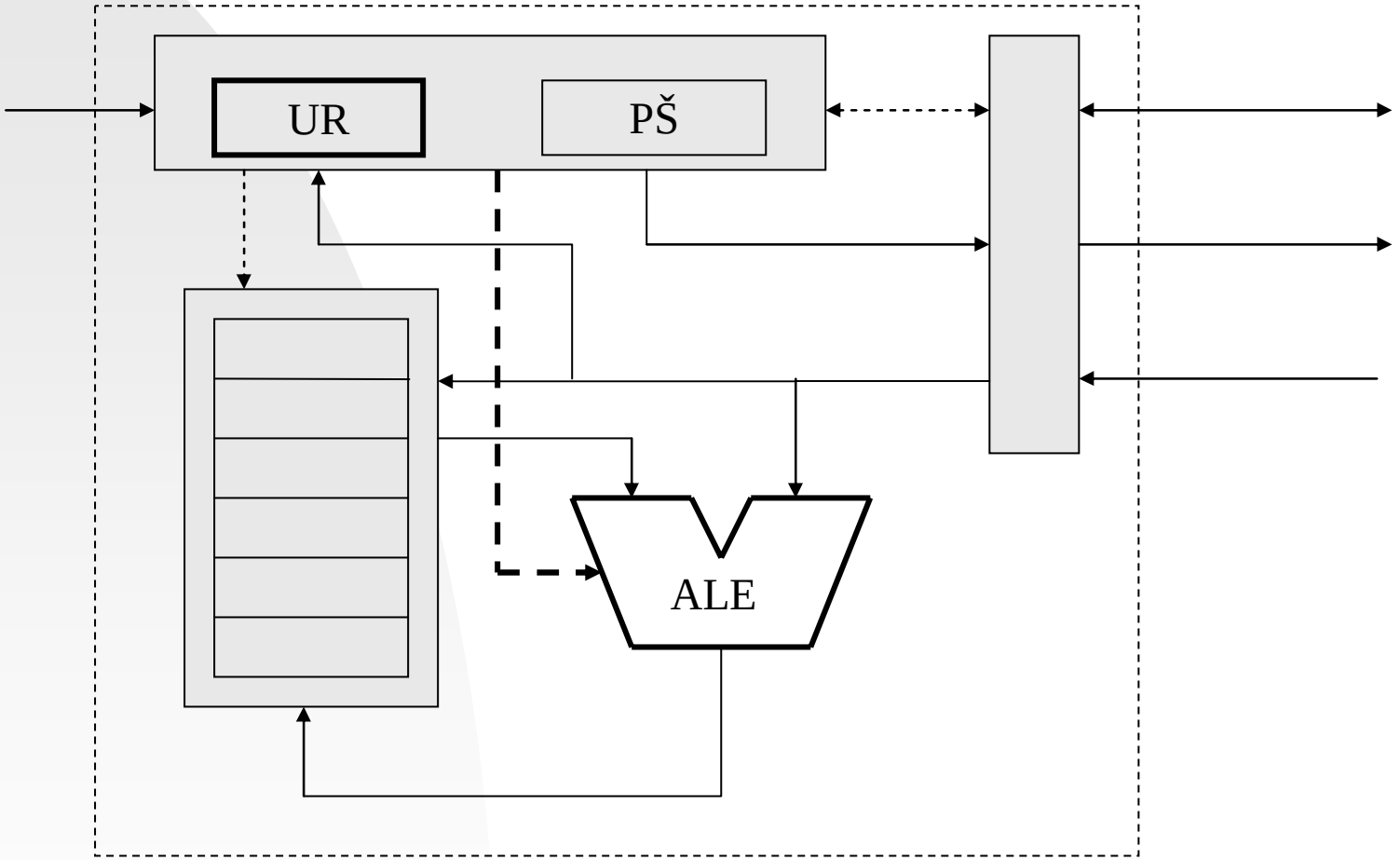
- Priprava parametrov



Priprava parametrov

- Skoraj vsi ukazi, ki jih mikroprocesor izvaja se nanašajo na določene podatke (parametre)
- Parametri so lahko v registrih, v zunanjem (podatkovnem) pomnilniku ali pa so del ukaza (kot konstante)
- V vsakem primeru mora krmilna enota poskrbeti, da se parametri prenesejo v notranjost mikroprocesorja

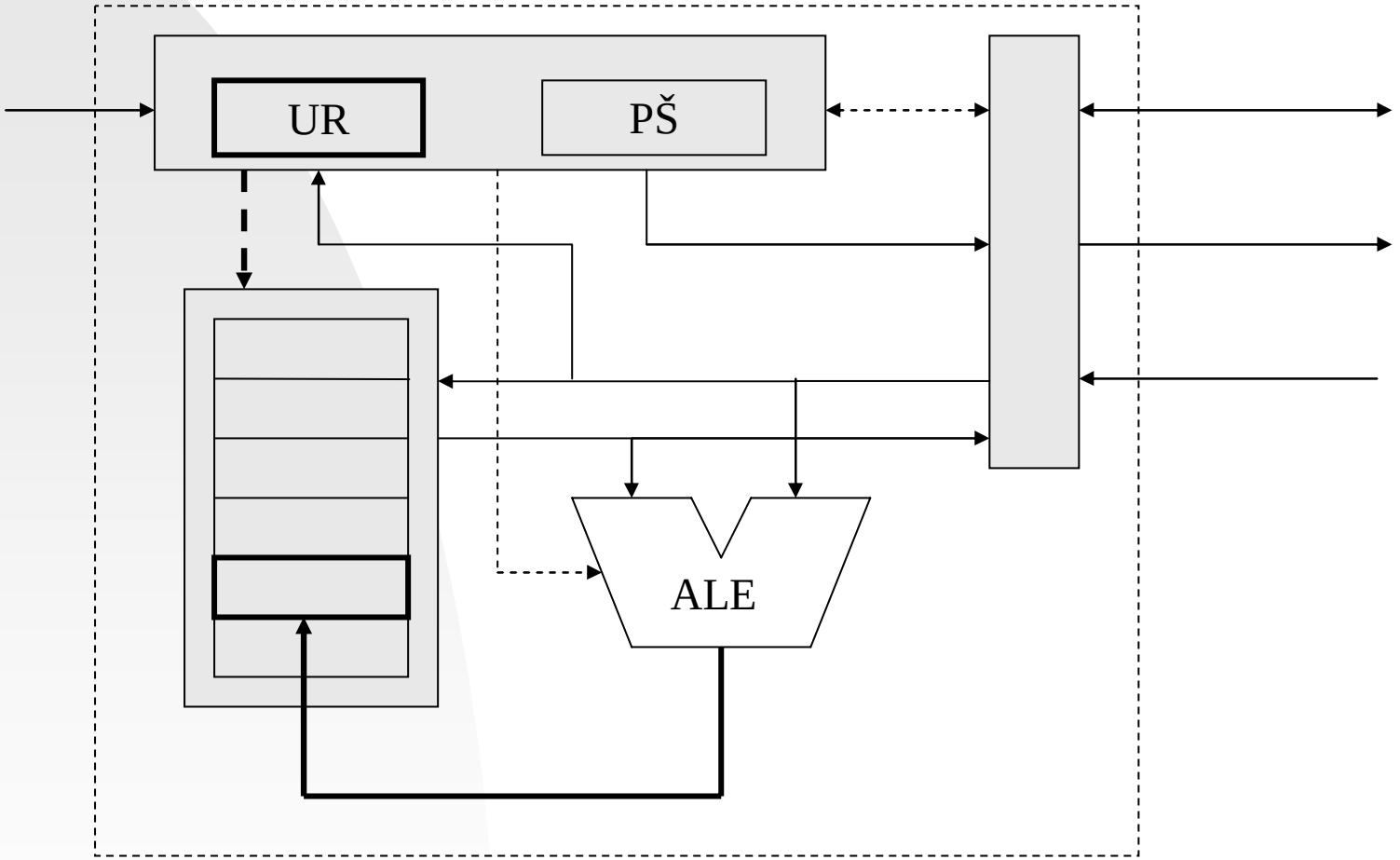
- Izvedba ukaza



Izvedba ukaza

- Običajno ukazi mikroprocesorja predstavljajo izvedbo neke aritmetične (matematične) ali logične operacije oz. premik nekega podatka iz enega mesta na drugega
- Za izvedbo aritmetičnih in logočnih ukazov poskrbi ALE

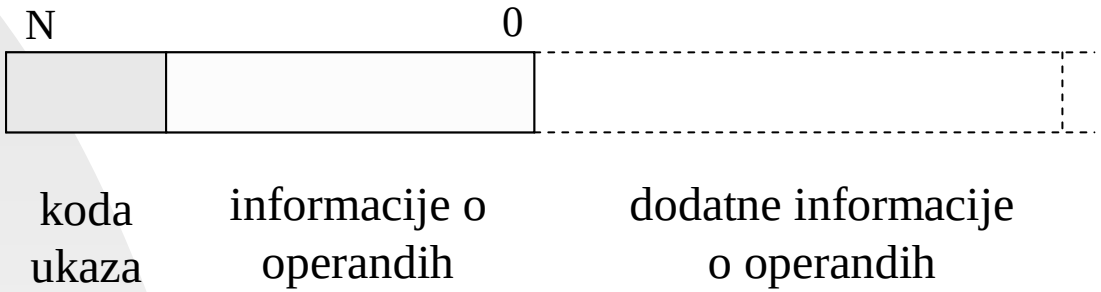
- Shranjevanje rezultatov



Shranjevanje rezultatov

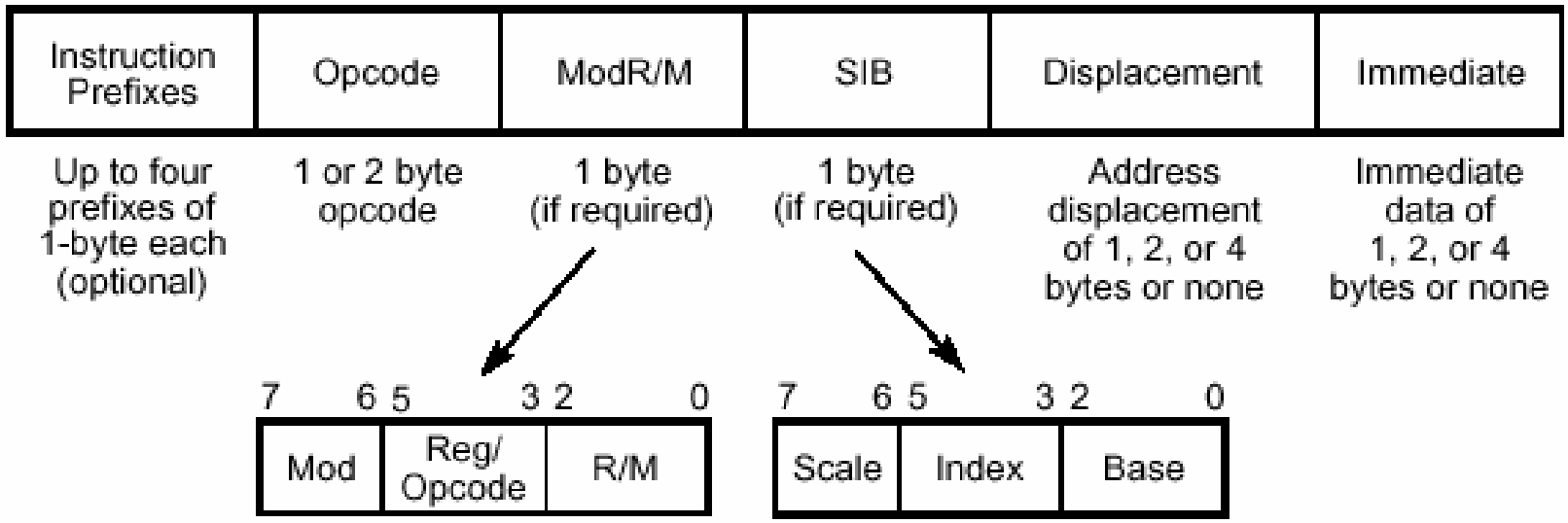
- Rezulta operacije mikroprocesorja je potrebno shraniti za nadaljno uporabo
- Rezulta operacije lahko shranimo ali v registrih ali v pomnilniku
- Določeni ukazi zavržejo rezultat operacije in shranijo samo informacijo o tem kakšna je bila vrednost rezultata (enaka nič, pozitivna, negativna, ...)

Oblika strojnih ukazov



- Ukazi spremenljive dolžine
  - Boljša izkoriščenost pomnilnika
  - Več ciklov za branje (izvedbo) daljših ukazov
- Ukazi fiksne dolžine
  - Slabša izkoriščenost pomnilnika
  - Celotni ukaz se prebere v enem ciklu

Ukazi spremenljive dolžine

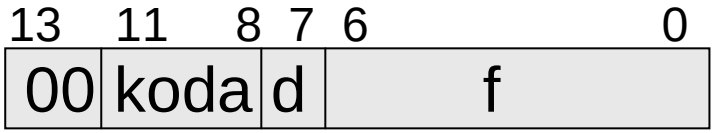


Format strojnih ukazov za Intelove procesorje

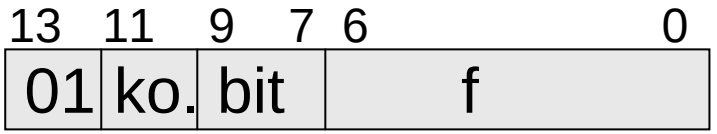
Ukazi fiksne dolžine

PIC

Registerski ukazi

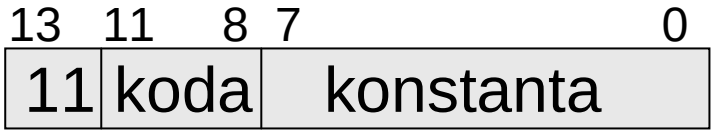


Ukazi nad biti

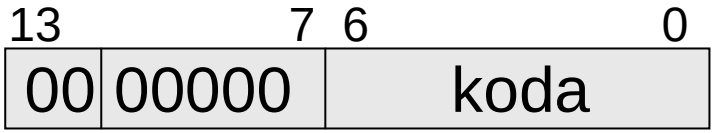


GOTO, CALL

Ukazi s konstanto

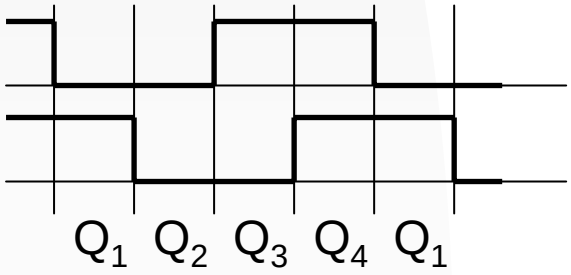


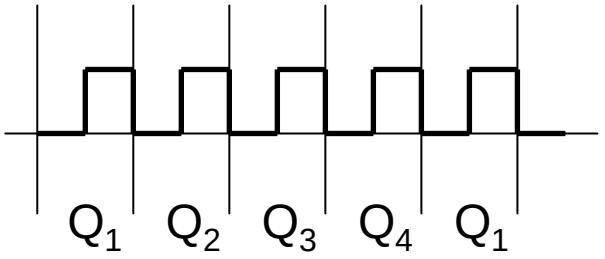
Posebni ukazi



Implementacija krmilne enote

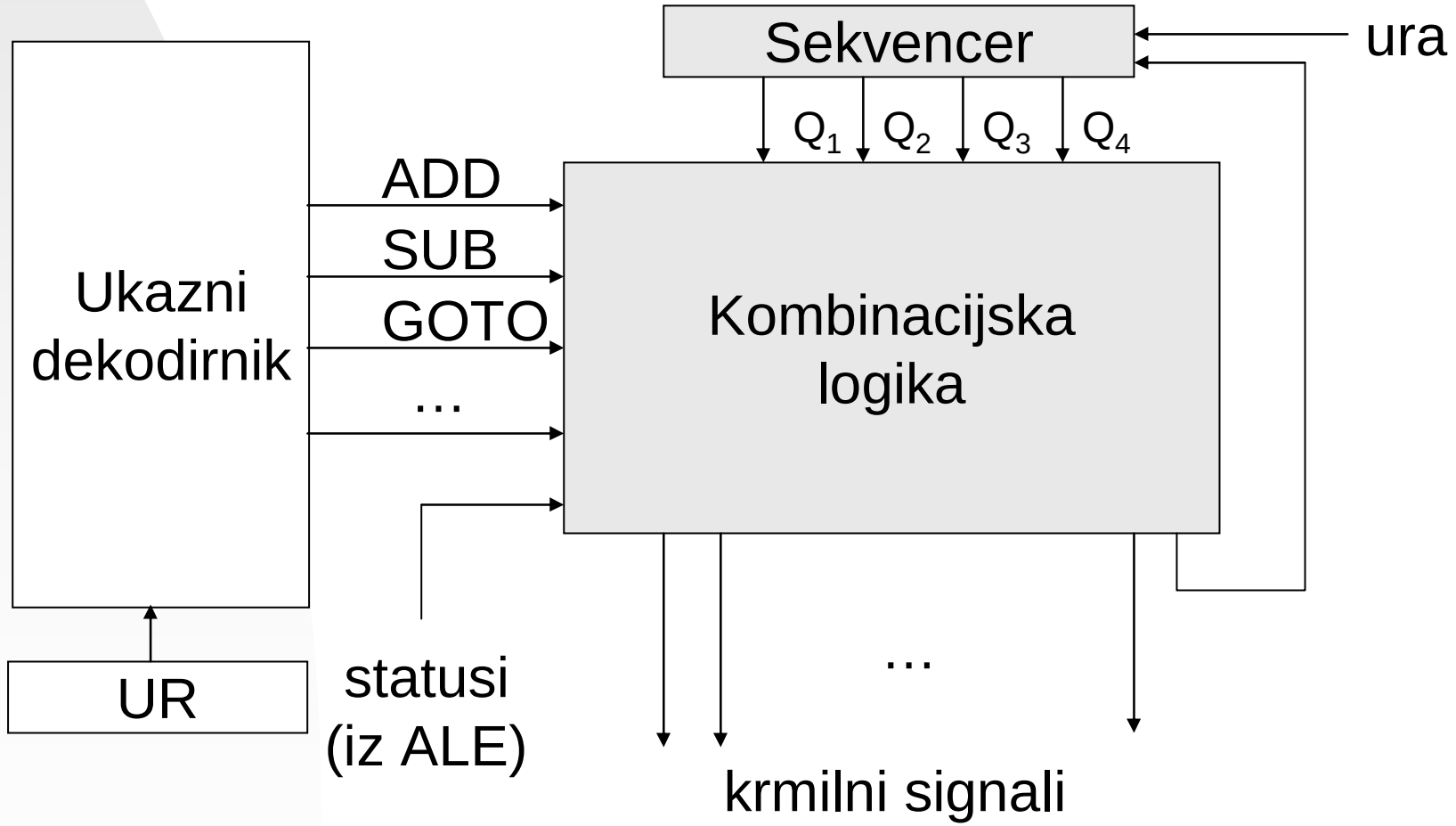
- Dve osnovni izvedbi:
  - kombinacijska logika
  - mikroprogramiranje
- Izvajanje ukazov poteka v taktu ure
  - več fazno zamaknjenih urinih signalov
  - več urinih taktov za izvedbo enega ukaza





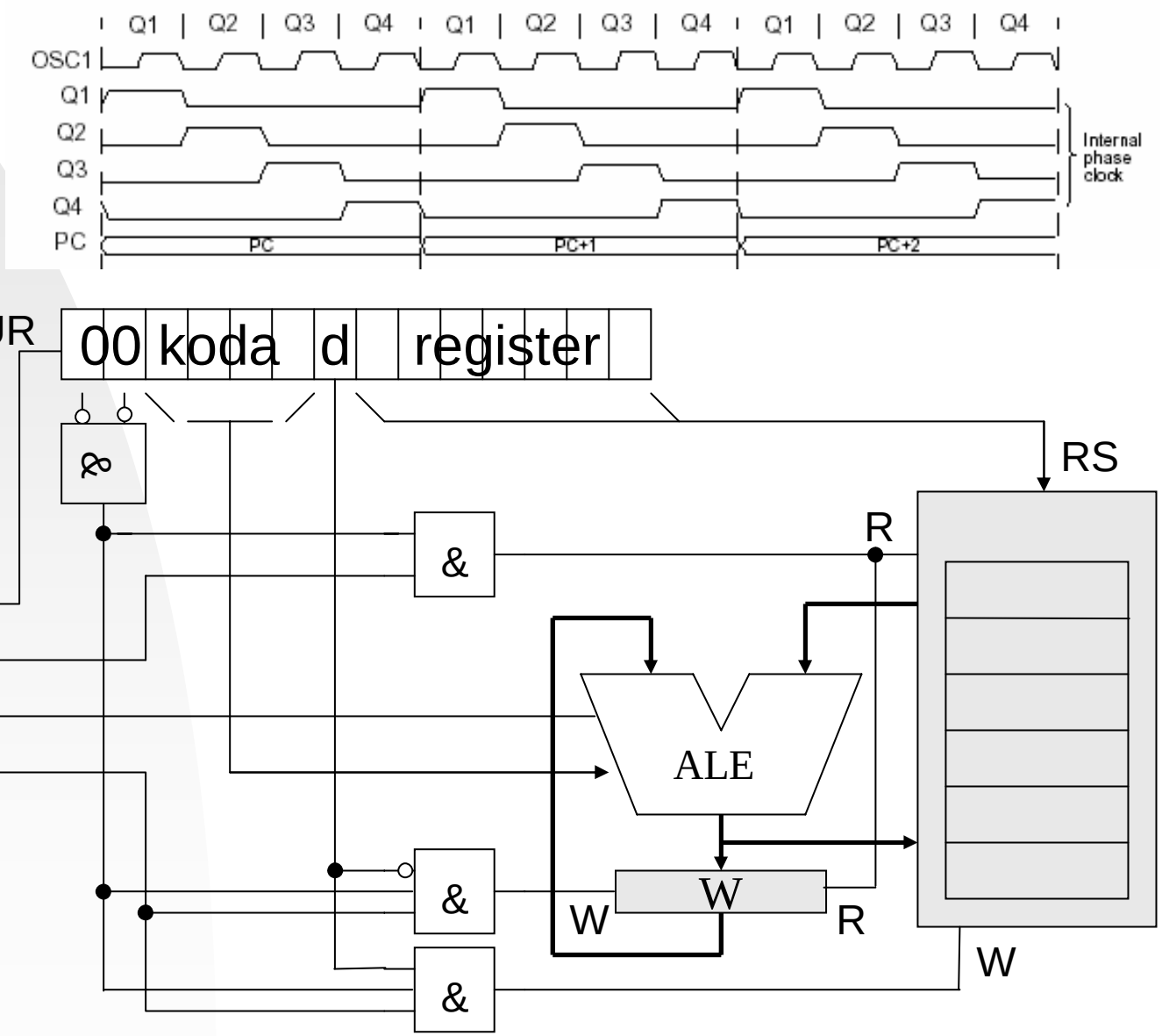
Krmilna enota s kombinacijsko logiko

- Dekodiranje in izvajanje ukazov poteka s pomočjo diskretnih logičnih vezij

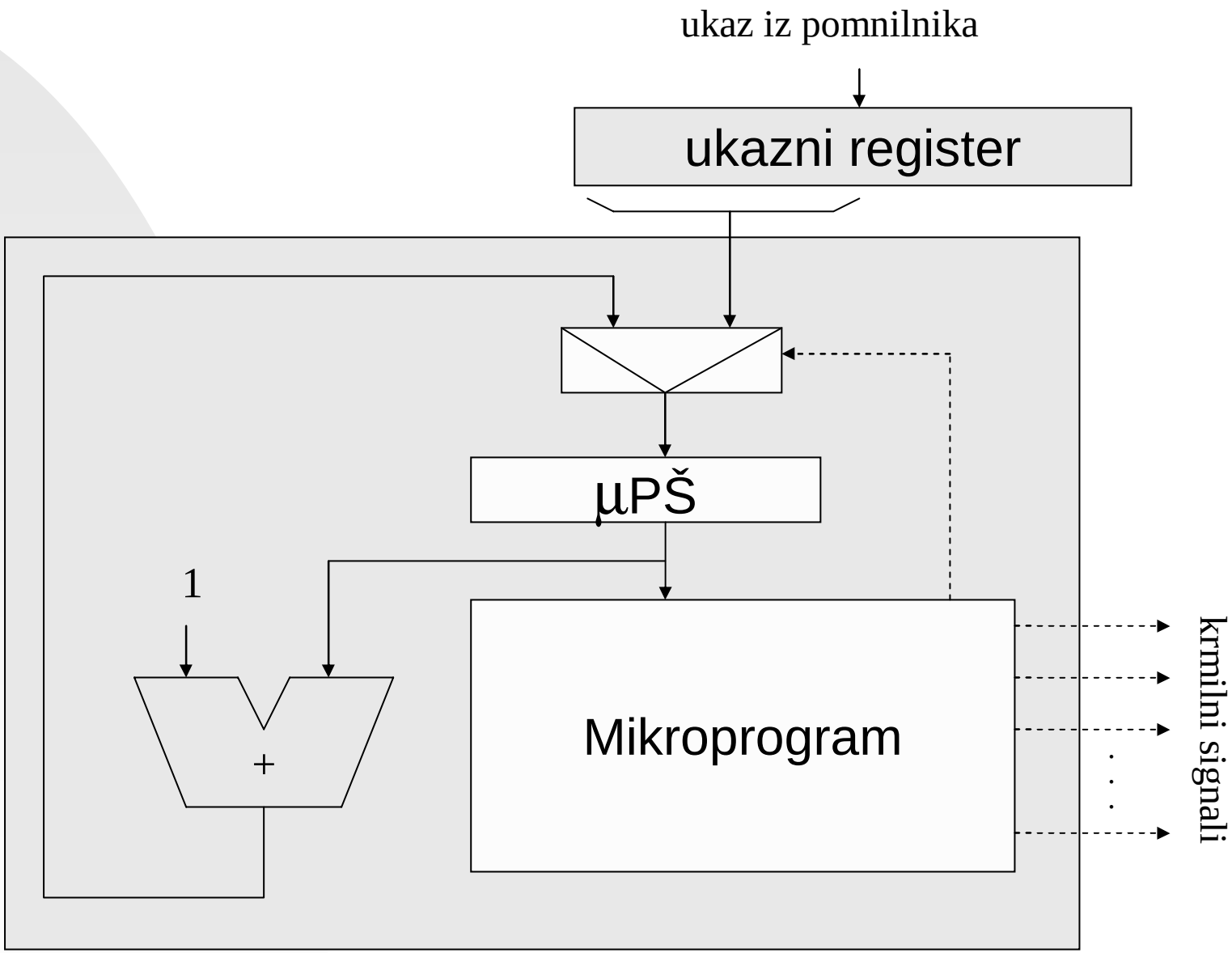


Zgled krmilne enote s kombinacijsko logiko

- PIC



Zgled mikroprogramirane krmilne enote



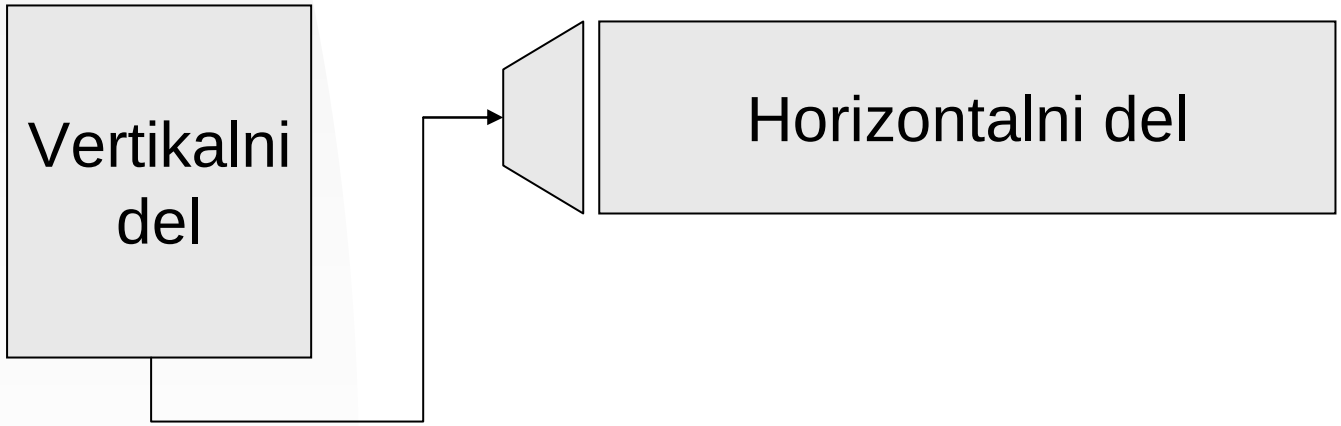
Različne izvedbe mikroprograma

- Horizontalna mikrokoda

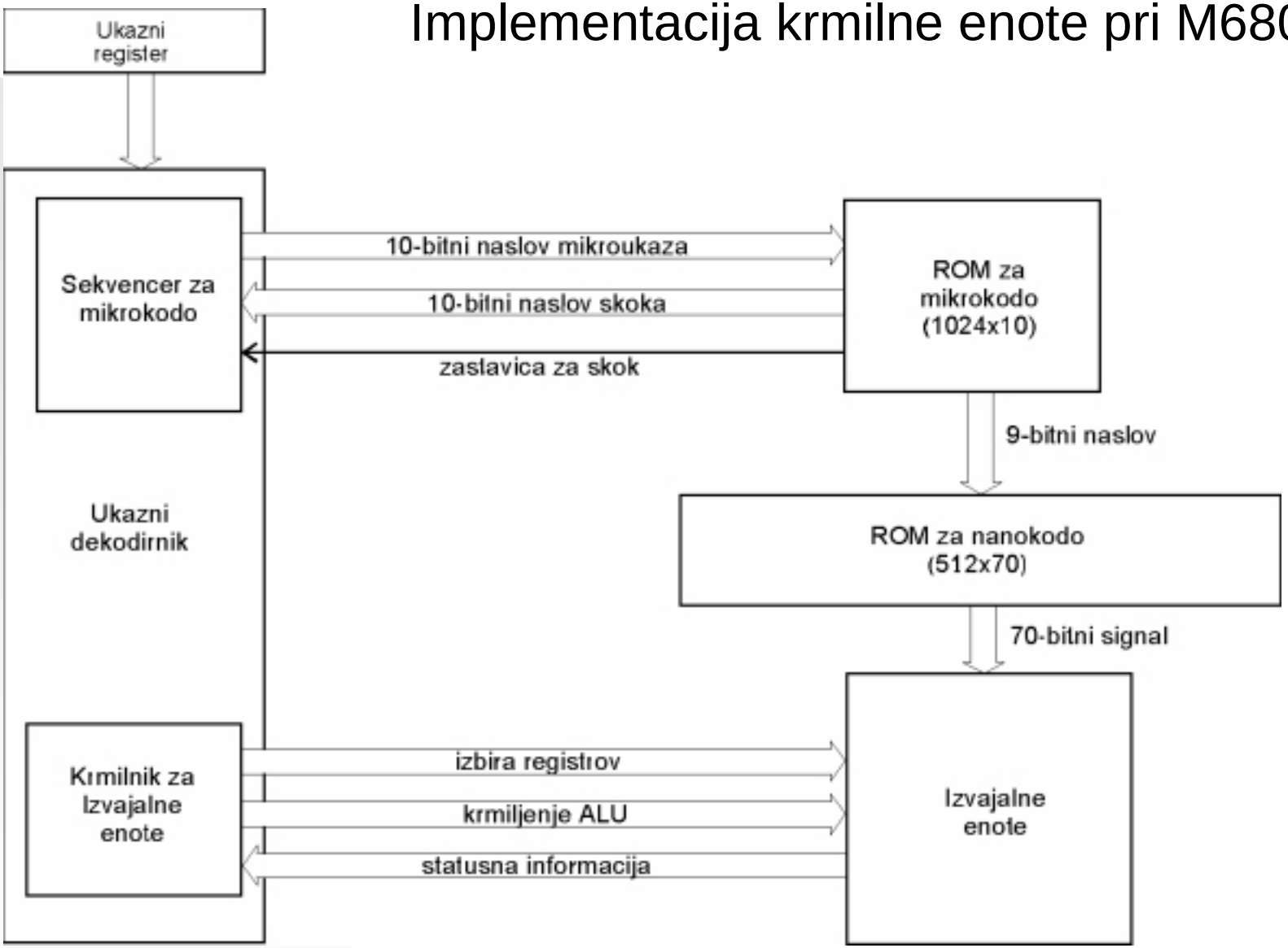
Za vsako enoto svoj nabor bitov

- Vertikalna mikrokoda

Kompaktne mikroinštrukcije z dodatni dekodiranjem



Implementacija krmilne enote pri M68000

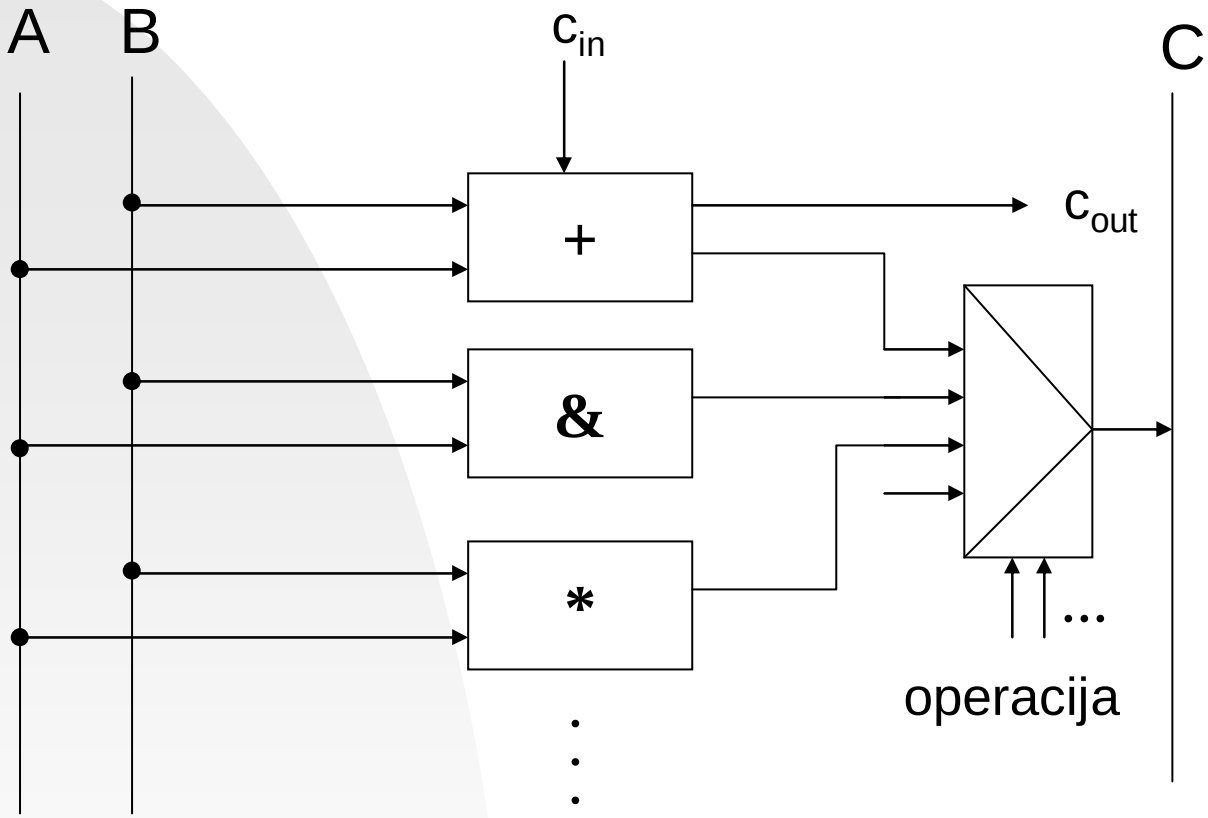


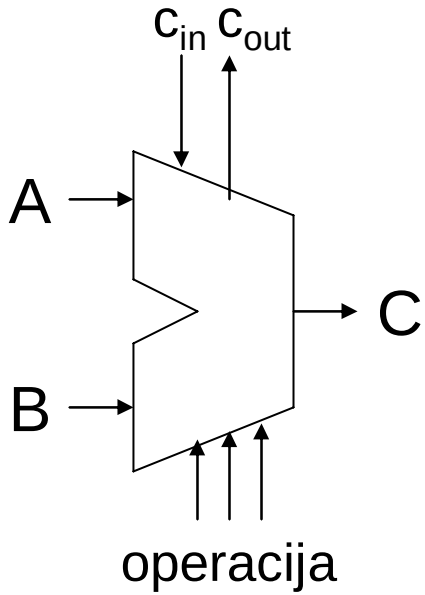
Oblika mikroukaza

Implementacija ALE

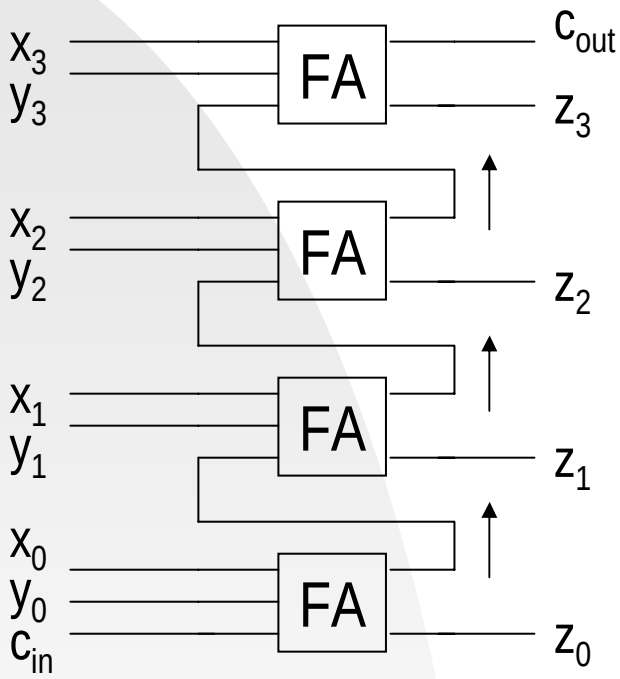
- ALE izvajaja vse operacije nad operandi (razen prenašanja informacij)
- Osnovne funkcije:
  - Aritmetični ukazi
  - Logočni ukazi
  - Delo z biti
  - Delo z nizi podatkov

- Splošna izvedba ALE



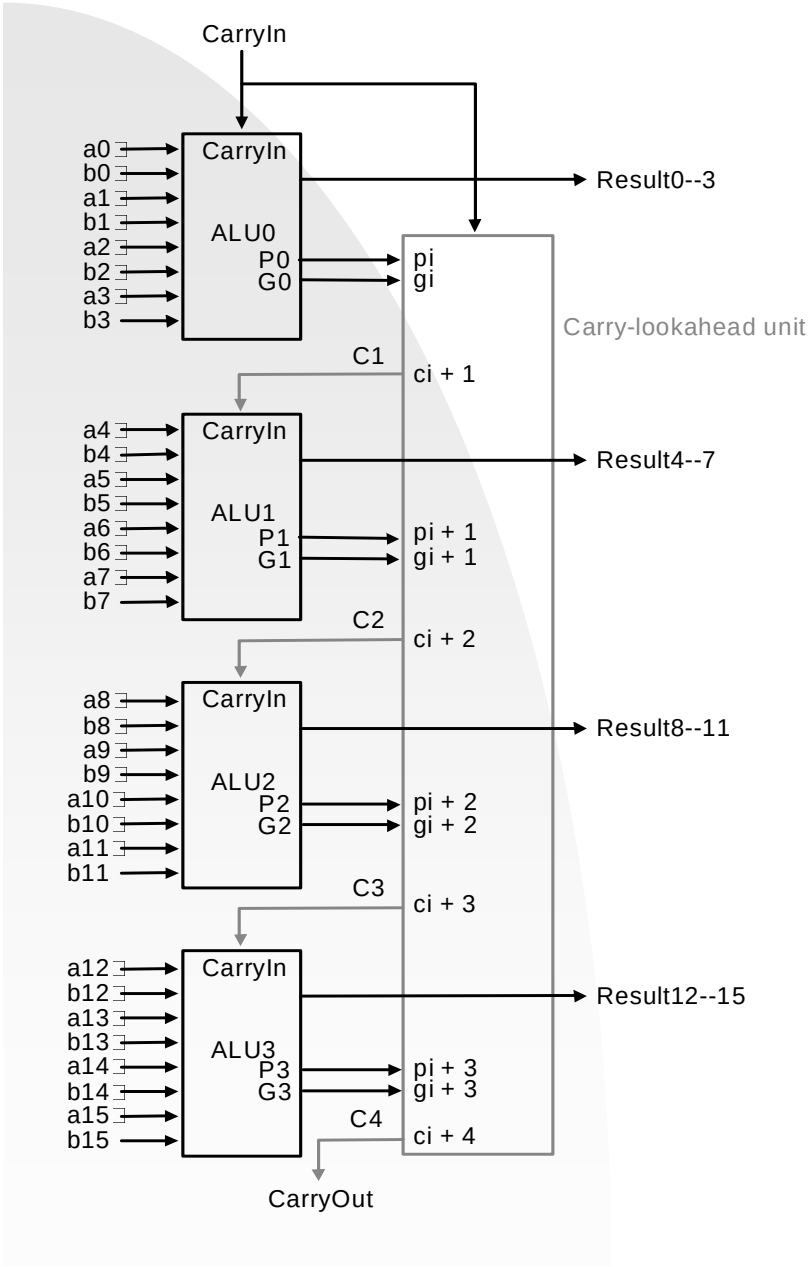


- Logične operacije
  - Implementirane so neposredno z navadnimi logičnimi vrati
  - Večjo širino besede dosežemo z vzporedno vezavo
- Celoštevilčno seštevanje
  - Pri seštevanju lahko pride do prenosa (carry – C bit) – rezultat ima večji število bitov kot je število bitov v osnovni besedi
  - Uporabimo polne seštevalnike vezane vzporedno in z upoštevanjem prenosa med posameznimi biti



- Izračun bita *i* zahteva predhodni izračun bita *i-1*

Pohitritev izračuna bita prenosa pri seštevanju



1 bitni seštevalnik:

$g_i=a_ib_i$ - primer, ko se vedno generira prenos

$p_i=a_i+b_i$ - primer, ko se generira prenos ob $c_{in}=1$

4 bitni seštevalnik:

$c_1 = g_0 + p_0c_0$

$c_2 = g_1 + p_1c_1 = g_1 + p_1g_0 + p_1p_0c_0$

$c_3 = g_2 + p_2c_2 = g_2 + p_2g_1 + p_2p_1g_0 + p_2p_1p_0c_0$

$c_4 = g_3 + p_3c_3 = ...$

16 bitni seštevalnik:

$P_i=p_3p_2p_1p_0$

$G_i=g_3+g_2p_3+g_1p_3p_2+g_0p_3p_2p_1$

$C_1=G_0+P_0C_0$

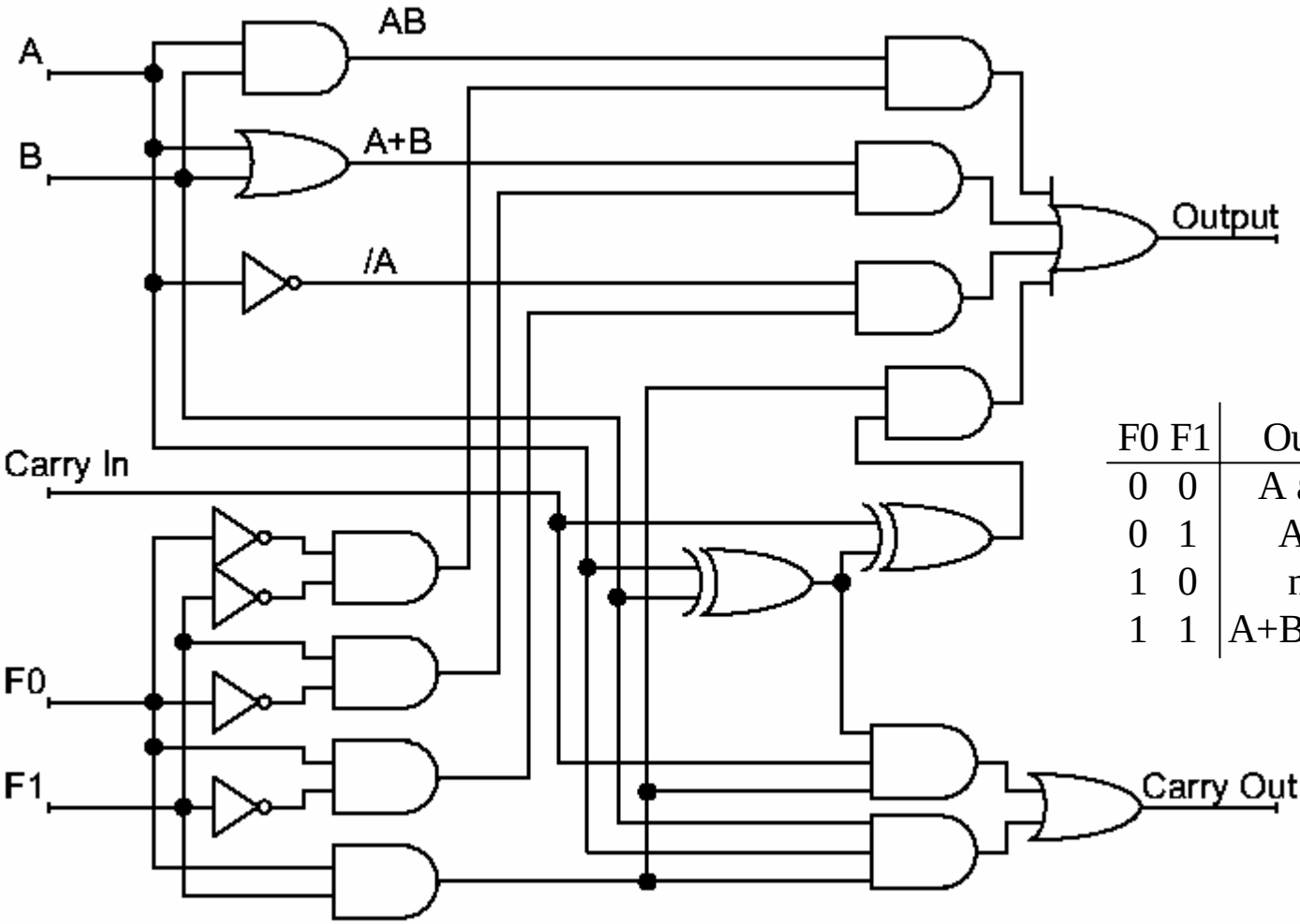
$C_2=G_1+P_1C_1$

$C_3=G_2+P_2C_2$

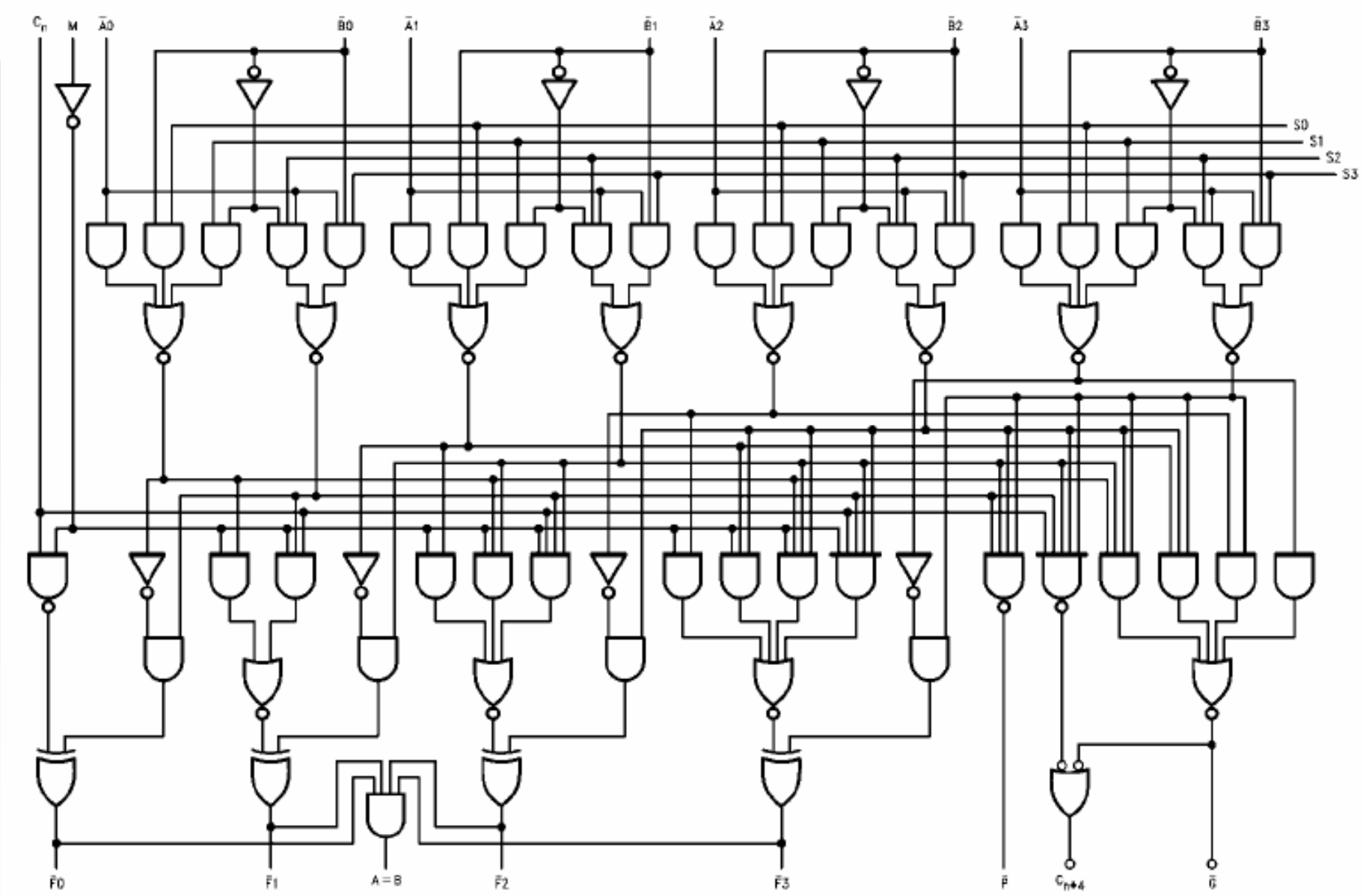
$C_4=G_3+P_3C_3$

- Celoštevilčno odštevanje
  - Prevedemo na prištevanje negativnega števila
  - Če za predstavitev negativnih števil uporabimo dvojiški komplement, lahko uporabimo ista vezja tako za predznačena kot nepredznačena
  - Negativna števila imajo najvišji bit enak 1 (N bit)
  - Pri delu z neprznačenimi števili se pojavi problem prekoračitve obsega – overflow (V bit)
    Nastane ko se bit prenosa v najbolj obteženi bit razlikuje od bita prenosa iz zadnjega:

$$V = C_n \oplus C_{n-1} \quad (n = \text{najbolj obteženi bit})$$



F0	F1	Output	CaryOut
0	0	A and B	0
0	1	A or B	0
1	0	not A	0
1	1	A+B+CaryIn	CaryOut



74LS181 4 bitna ALE

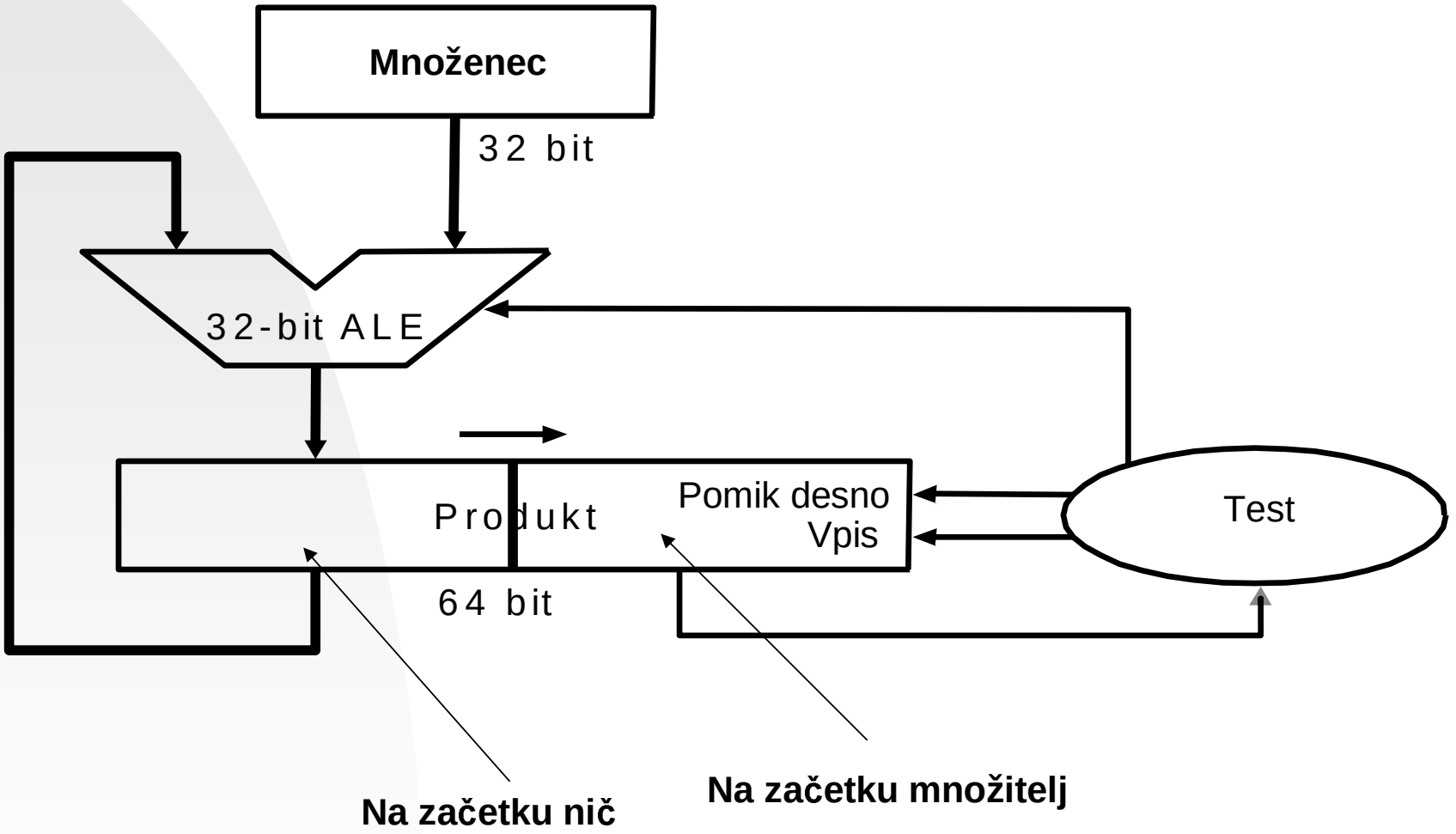
Mode Select Inputs				Active LOW Operands & F_n Outputs		Active HIGH Operands & F_n Outputs	
				Logic	Arithmetic (Note 2)	Logic	Arithmetic (Note 2)
S3	S2	S1	S0	(M = H)	(M = L) (C_n = L)	(M = H)	(M = L) (C_n = H)
L	L	L	L	$\overline{A}$	A minus 1	$\overline{A}$	A
L	L	L	H	$\overline{AB}$	AB minus 1	$\overline{A} + \overline{B}$	A + B
L	L	H	L	$\overline{A} + \overline{B}$	$A\overline{B}$ minus 1	$\overline{A}$ B	A + $\overline{B}$
L	L	H	H	Logic 1	minus 1	Logic 0	minus 1
L	H	L	L	$\overline{A} + \overline{B}$	A plus (A + $\overline{B}$)	$\overline{AB}$	A plus $A\overline{B}$
L	H	L	H	$\overline{B}$	AB plus (A + $\overline{B}$)	$\overline{B}$	(A + B) plus $A\overline{B}$
L	H	H	L	$\overline{A} \oplus \overline{B}$	A minus B minus 1	A ⊕ B	A minus B minus 1
L	H	H	H	A + $\overline{B}$	A + $\overline{B}$	$A\overline{B}$	AB minus 1
H	L	L	L	$\overline{A}$ B	A plus (A + B)	$\overline{A}$ + B	A plus AB
H	L	L	H	A ⊕ B	A plus B	$\overline{A} \oplus \overline{B}$	A plus B
H	L	H	L	B	$A\overline{B}$ plus (A + B)	B	(A + $\overline{B}$) plus AB
H	L	H	H	A + B	A + B	AB	AB minus 1
H	H	L	L	Logic 0	A plus A (Note 1)	Logic 1	A plus A (Note 1)
H	H	L	H	$A\overline{B}$	AB plus A	A + $\overline{B}$	(A + B) plus A
H	H	H	L	AB	$A\overline{B}$ minus A	A + B	(A + $\overline{B}$) plus A
H	H	H	H	A	A	A	A minus 1

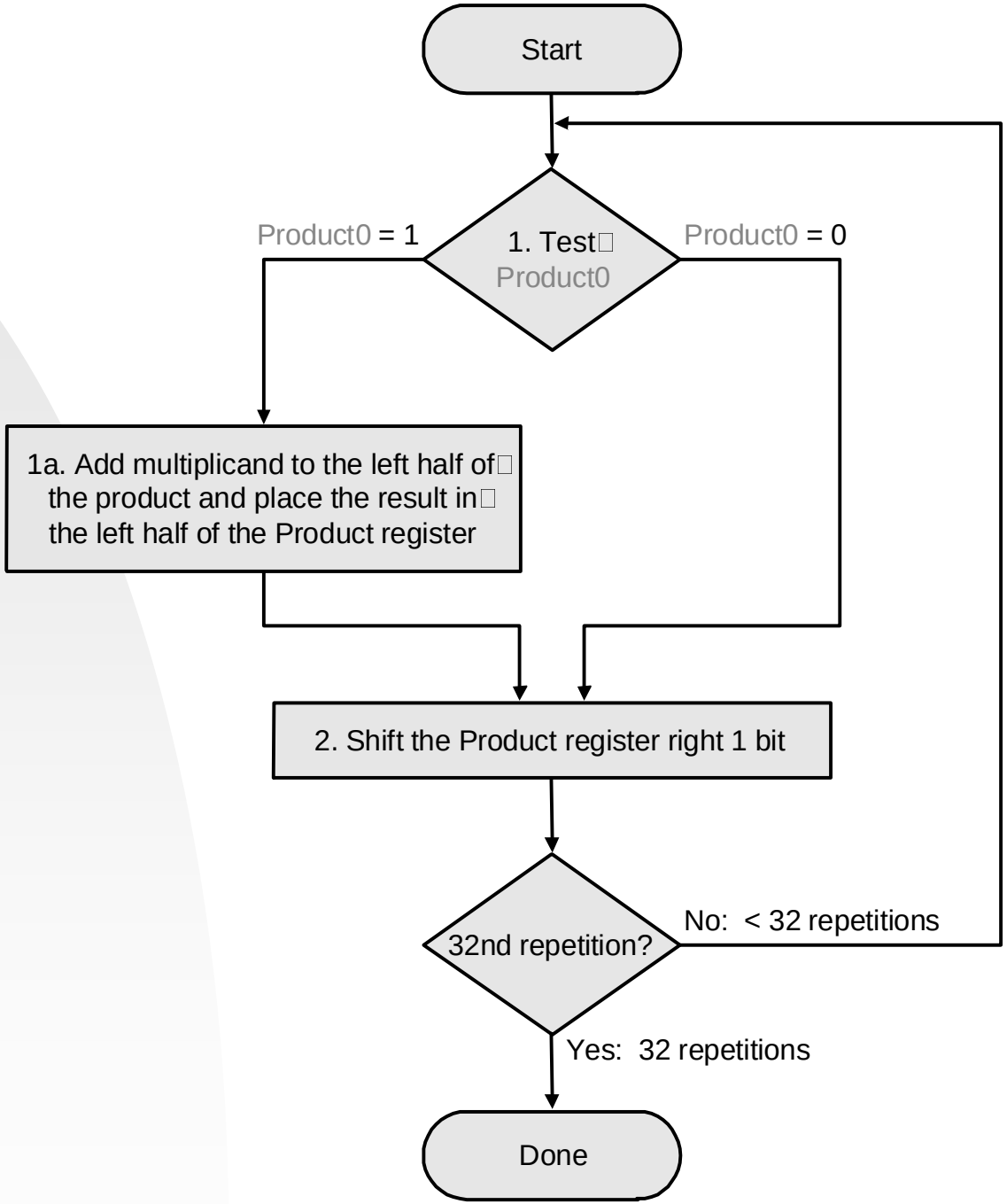
Note 1: Each bit is shifted to the next most significant position.

Note 2: Arithmetic operations expressed in 2s complement notation.

- Celostevilčno množenje
  - Prevedemo na zaporedno seštevanje
  - Uporabimo osnovnošolski “algoritem”

```
     0010
__x_1011
     0010
    0010
   0000
  0010___
 0010110
```


Množenec
32 bit
32-bit ALE
Produkt
Pomik desno
Vpis
Test
64 bit
Na začetku nič
Na začetku množitelj


Start
1. Test Product0
Product0 = 1
Product0 = 0
1a. Add multiplicand to the left half of the product and place the result in the left half of the Product register
2. Shift the Product register right 1 bit
32nd repetition?
No: < 32 repetitions
Yes: 32 repetitions
Done

- Za zmanjšanje števila operacij lahko uporabimo t.i. Boothov algoritem

Niz 0 v množitelju => brez sprememb produkta (samo pomik)
Niz 1 v množitelju => sprememba samo pri prvi in po zadnji enici

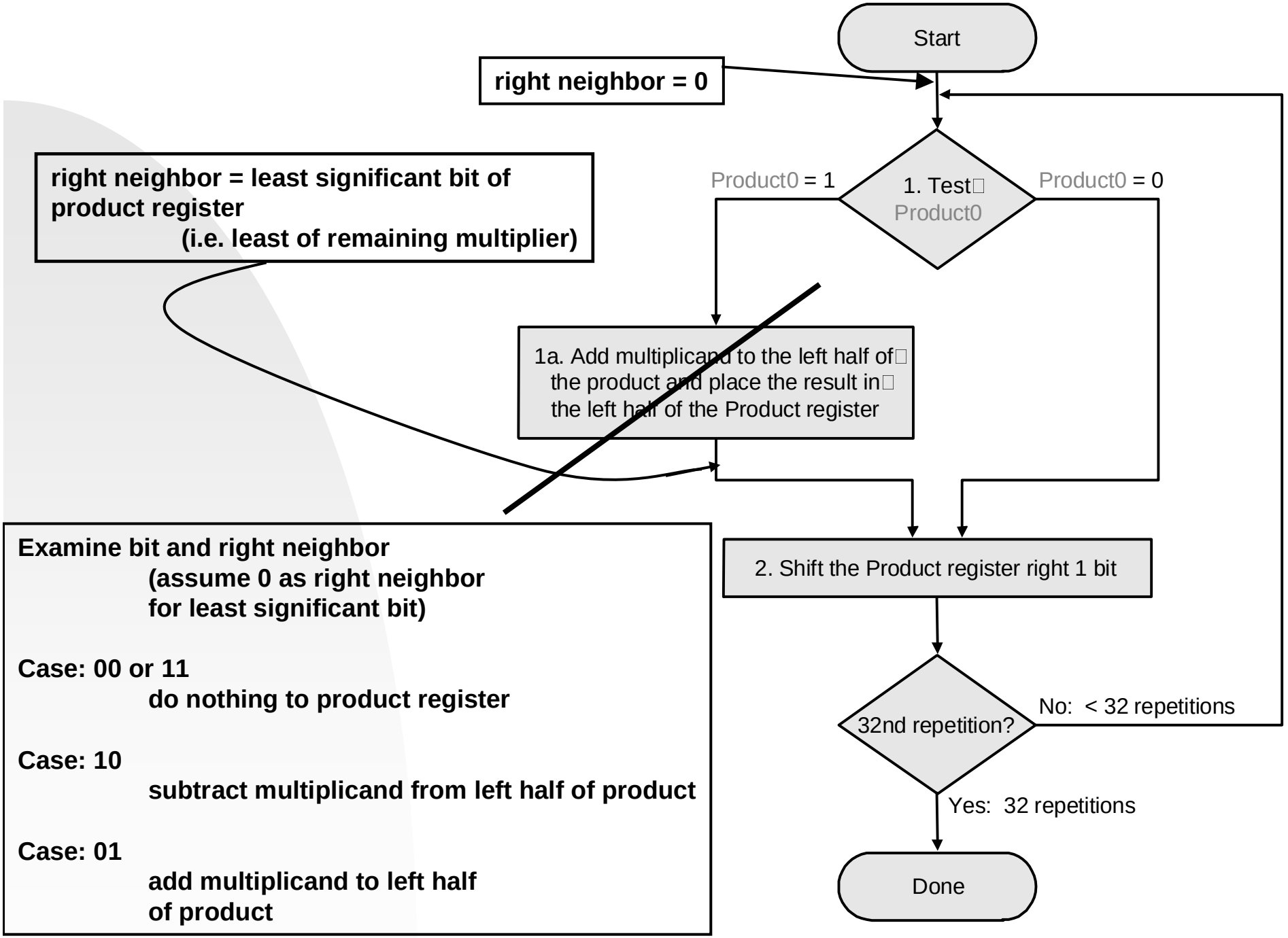
Zgled: množitelj = 0010 1111 1101 1100

produkt popravimo v teh točkah

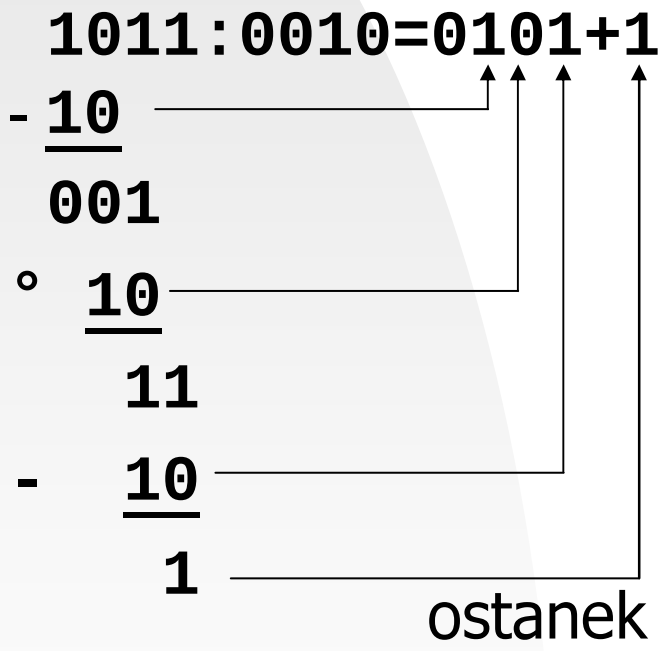
Ideja: 0010 1111 1101 1100
1 0000 00

2^k-1 zamenjamo z 2^k $(2^k-1)*X = 2^k*X-X$

Pri prvi 1 (z desne) množenec odštejemo
Po zadnji 1 (z desne) množenec prištejemo

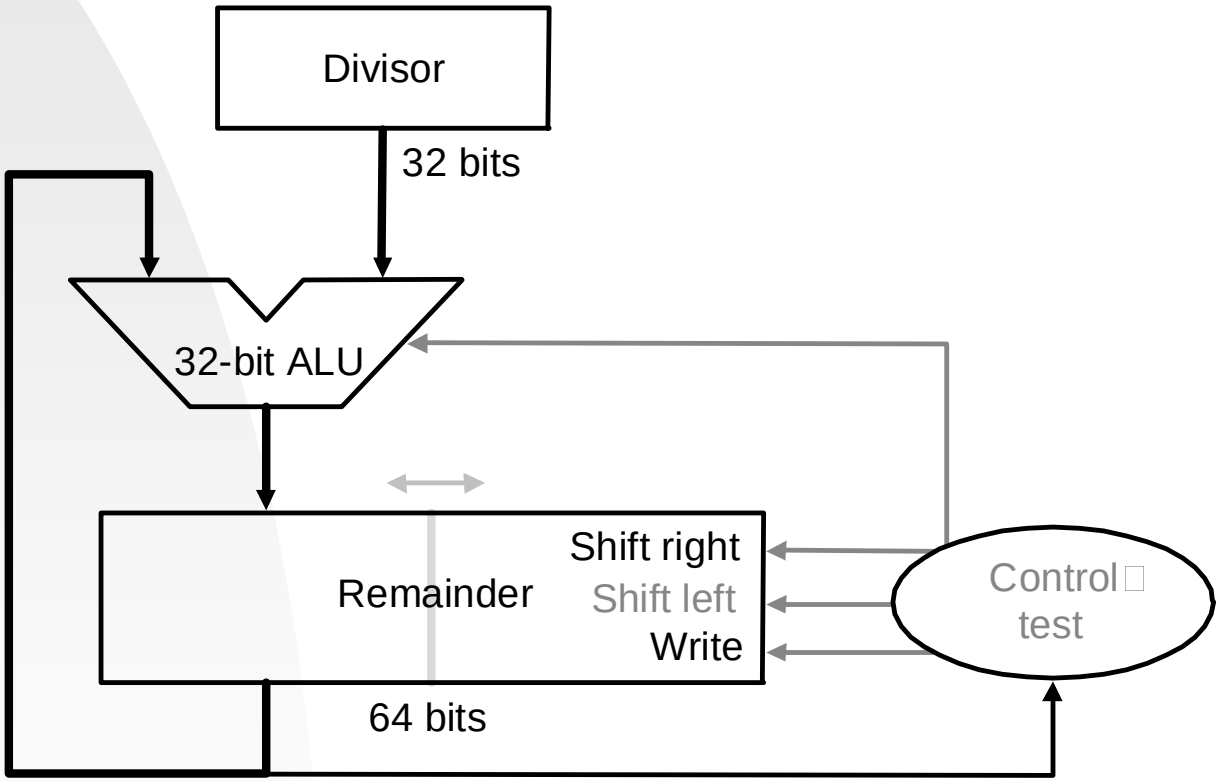

Start
right neighbor = 0
1. Test Product0
Product0 = 1
Product0 = 0
right neighbor = least significant bit of product register
(i.e. least of remaining multiplier)
1a. Add multiplicand to the left half of the product and place the result in the left half of the Product register
2. Shift the Product register right 1 bit
32nd repetition?
No: < 32 repetitions
Yes: 32 repetitions
Done
Examine bit and right neighbor
(assume 0 as right neighbor
for least significant bit)
Case: 00 or 11
do nothing to product register
Case: 10
subtract multiplicand from left half of product
Case: 01
add multiplicand to left half
of product

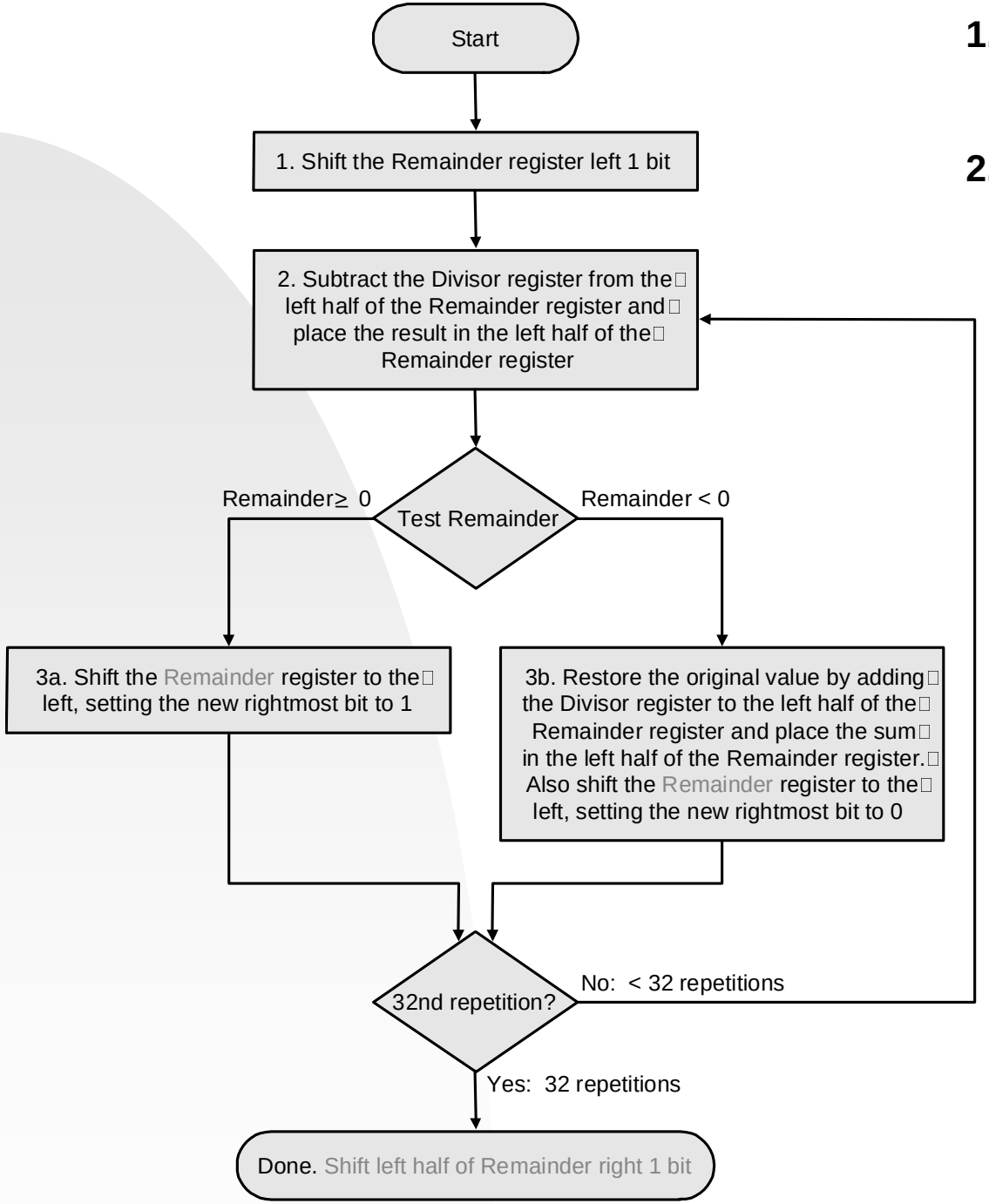
- Celoštevilčno deljenje
  - Prevedemo na zaporedno odštevanje
  - Uporabimo “osnovnošolsko” aritmetiko



1. Prvo enico delitelja podpišemo pod deljenec
2. Primerjamo z deljencem (ostankom)
   Če je delitelj večji, odštejemo in zapišemo 1, sicer zapišemo 0
3. Premaknemo delitelj za eno mesto v desno

- Celoštevilčno deljenje
  - V spodnjo in zgornjo polovico ostanka prepišemo deljenec





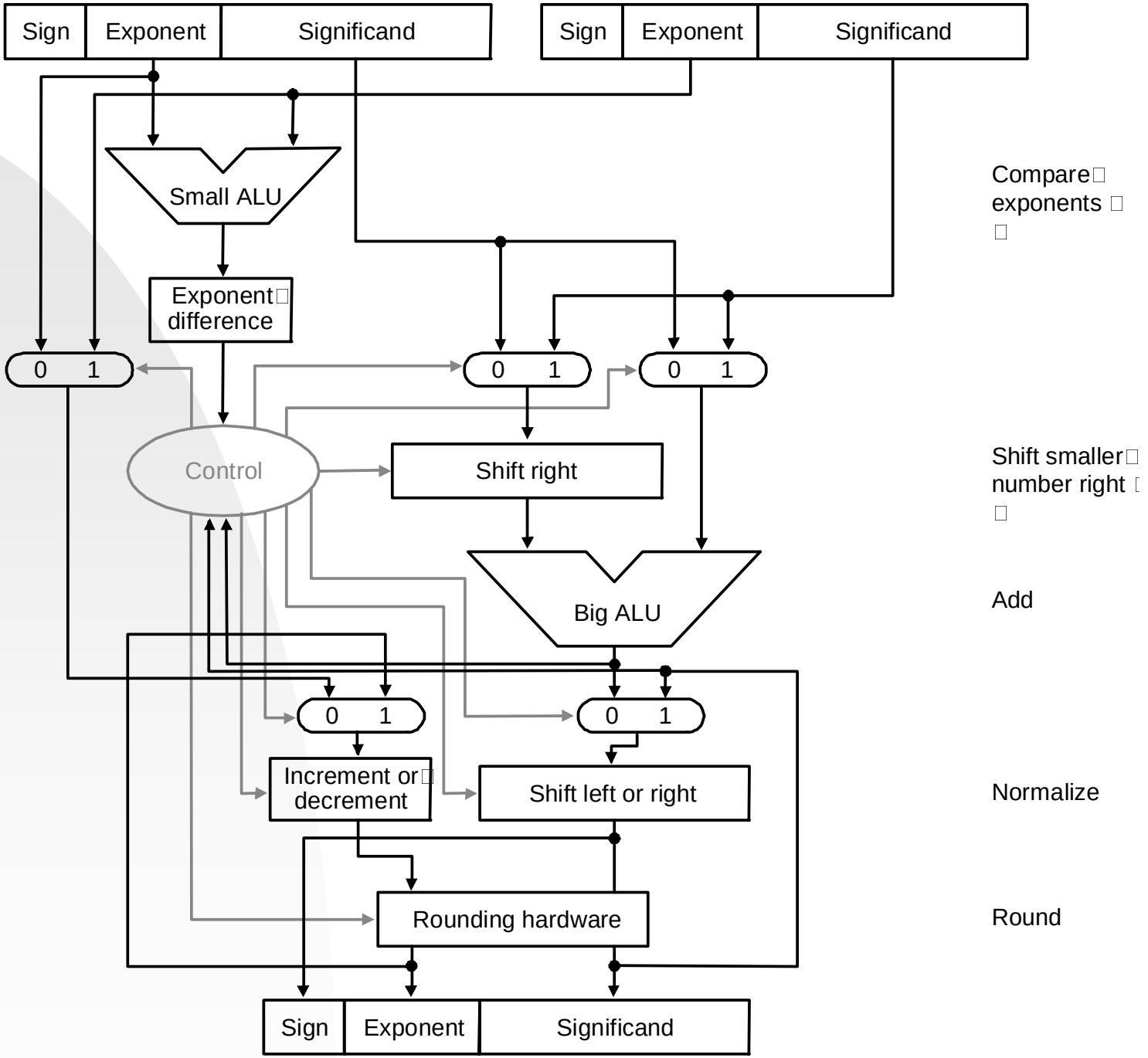
1. **Change to shift before subtract (save one iteration)**

2. **can avoid restoration step**
   r = remainder portion
   d = divisor

   Note: 2(r + d) - d = 2r + d =>

   can *add* d on next step to obtain require bit pattern

- Operacije nad realnimi števili
  - Realno število prevedemo na predznak, mantiso in eksponent
    $X = (-1)^P * M * 2^E$
  - Operacije izvajamo na mantisi in eksponentu hkrati
  - Pri določenih operacijah (+, -) je potrebno oba operanda najprej spraviti na isti eksponent
  - Po izvedbi operacije je potrebno rezultat pretvoriti v normalizirano obliko
  - Običajno se uporablja standard IEEE 754:
    - Enojna natančnost: 8 bitni eksponent, 23 bitna mantisa
    - Dvojna natančnost: 11 bitni eksponent, 53 bitna mantisa


Sign
Exponent
Significand
Sign
Exponent
Significand
Small ALU
Exponent difference
0 1
0 1
0 1
Control
Shift right
Big ALU
0 1
0 1
Increment or decrement
Shift left or right
Rounding hardware
Sign
Exponent
Significand
Compare exponents
Shift smaller number right
Add
Normalize
Round

- Kompleksnejše operacije nad realnimi števili
  - razvoj funkcij v vrsto – vsak dodani člen poveča natančnost rezultata

    $\sin(x)=1-x^3/3!+x^5/5!-x^7/7!+\ldots$

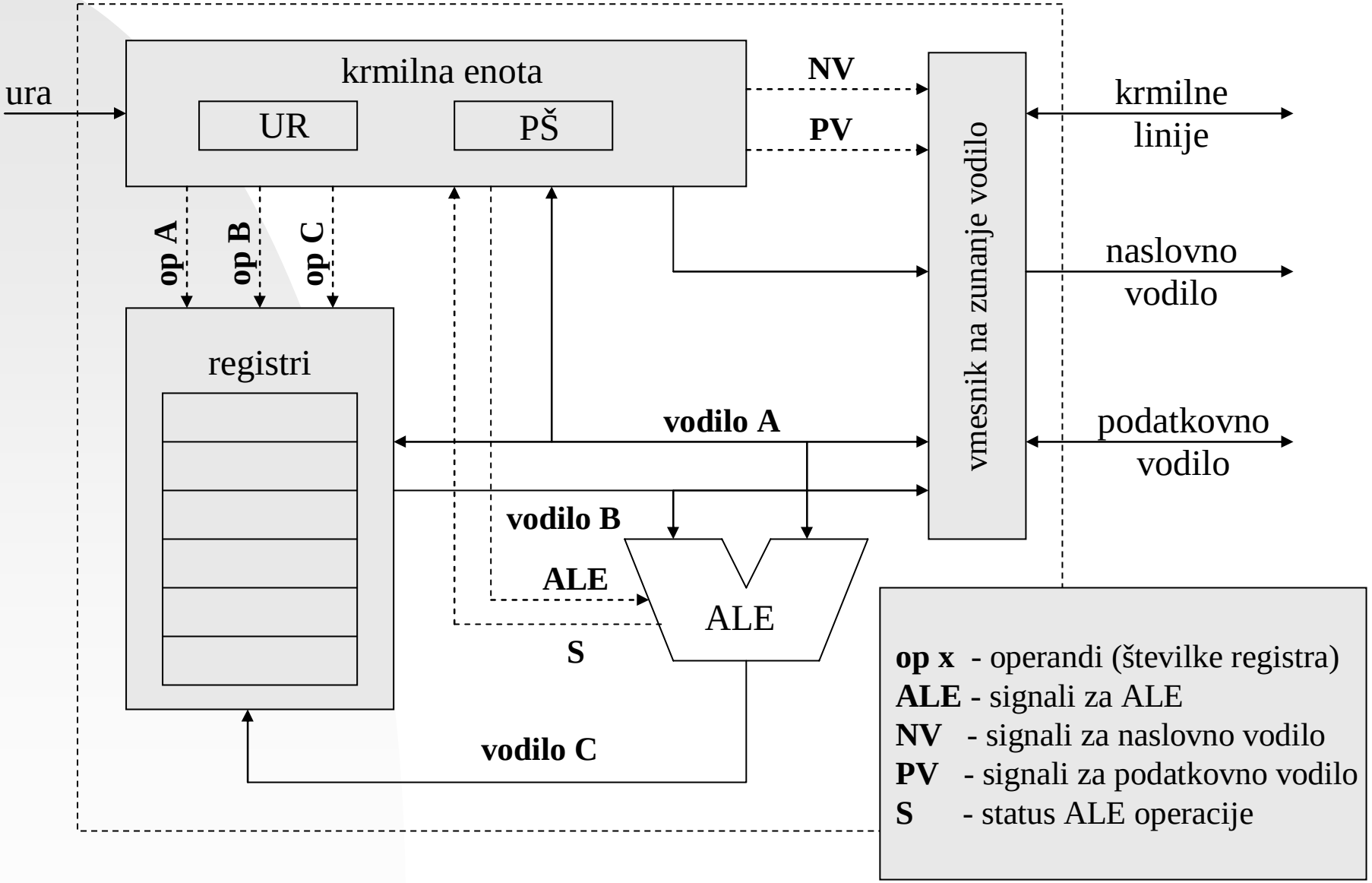
    http://math.furman.edu/~dcs/java/taylor.html

  - posebni algoritmi

    za $X=\sqrt{A}$

    iterativna formula: $X_{n+1}=(X_n-A/X_n)/2$

Zgled delovanja hipotetičnega mikroprocesorja



Nabor ukazov

0000 - ADD a,b,c	$R_C = R_A + R_B$
0001 - SUB a,b,c	$R_C = R_A - R_B$
0010 - OR a,b,c	$R_C = R_A$ or R_B
0011 - AND a,b,c	$R_C = R_A$ and R_B
0100 - NEG a,b	$R_C = -R_A$
0101 - NOT a,b	R_C = not R_A
0110 - MOV [a],b	$R_C = [R_A]$
0111 - MOV a,[b]	$[R_A] = R_B$

[x] - vsebina pomnilnika

1000 - MOV naslov,A	$[n] = R_A$
1001 - MOV A,naslov	$R_A = [n]$
1010 - MOV #konst,A	$R_A = k$

MOV a,b ≡ OR a,a,b

1100	00 - JMP n	PŠ = n
1100	01 - JZ n	if ZERO then PŠ = n
1100	10 - JC n	if CARY then PŠ = n

1110 - JMP [a]	PŠ = R_A
1111 - MOV PC,a	R_A = PŠ

Zgradba mikro ukaza

PŠ
- 00 - brez sprememb
- 01 - povečaj za 1
- 10 - naloži iz UR
- 11 - naloži iz vodila A

ALE
- 000 - brez sprememb
- 001 - C = A
- 010 - C = A+B
- 011 - C = A-B
- 100 - C = A or B
- 101 - C = A and B
- 110 - C = not A
- 111 - C = -A

A,B,C
- 0 - neaktivno
- 1 - dostop do registra (A in B branje, C pisanje)

UR
- 0 - brez sprememb
- 1 - naloži iz vodila A

PV
- 00 - brez sprememb
- 01 - branje (na notranje vodilo A)
- 10 - pisanje (iz notranjega vodila A)
- 11 - pisanje (iz notranjega vodila B)

NV
- 00 - brez sprememb
- 01 - vsebina PŠ
- 10 - vsebina vodila A

številke registrov so določene z polji op A, op B in op C iz ukaza

Izvajanje ukaza

0 0 0 0 0 0 0 0 0 1 0 1 1 0 0 0 ADD R0,R5,R8

1.	0 0	0 0 0	0	0	0	0	0 0	0 1	vsebina PŠ se prenese na naslovno vodilo
2.	0 1	0 0 0	0	0	0	1	0 1	0 0	vrednost is podatkovnega vodila se prepiše v UR, dekodiranje ukaza, PŠ = PŠ+1
3.	0 0	0 1 0	1	1	1	0	0 0	0 0	na vodilo A se prenese R0, na vodilo B se prenese R5, ALE izvede operacijo seštevanja, rezultat se shrani v R8
1.	0 0	0 0 0	0	0	0	0	0 0	0 1	vsebina PŠ se prenese na naslovno vodilo
2.	0 1	0 0 0	0	0	0	1	0 1	0 0	vrednost is podatkovnega vodila se prepiše v UR, dekodiranje ukaza, PŠ = PŠ+1

Vprašanja

- Katere so osnovne komponente mikroprocesorja? Kakšno vlogo imajo?
- Kako v splošnem deluje mikroprocesor?
- Kakšne so možne oblike strojnih ukazov? Kakšne prednosti ima ena oblika pred drugo?
- Kakšne so možne izvedbe krmilne enote? Kakšne prednosti ima ena izvedba pred drugo?
- Kakšna je vloga ALE? Katere operacije izvaja?
- Na kakšen način ALE izvaja celoštevilčno seštevanje in odštevanje?
- Po kakšnih principih mikroprocesorji izvajajo kompleksnejše računske operacije s celimi in realnimi števili?