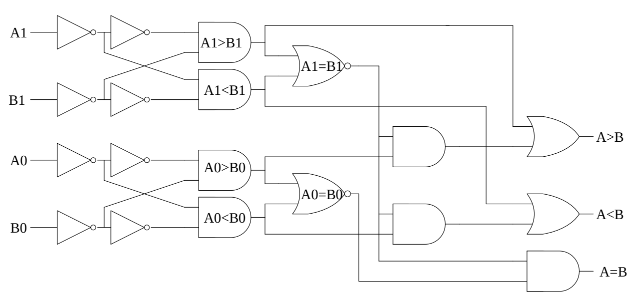
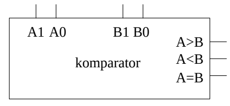
ARITMETIČKE OPERACIJE

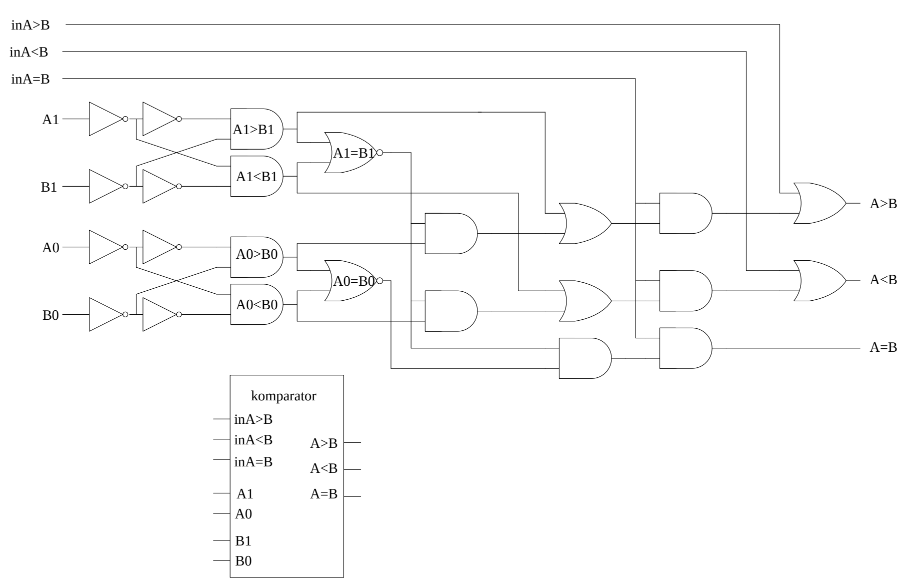
Poređenje - Komparatori

Postoje standardne kombinacione mreže binarni komparatori koji kao ulaze primaju dva binarna broja $A\ i\ B$, a na izlazima daju rezultat poređenja $A < B, A = B\ i\ A > B$. Isto kao i kod kodera prioriteta moguće je funkcije pojedinih izlaza minimizirati, međutim i komercijalno raspoložive komponente a i ovde ćemo prikazati komponentu kod koje je lako pratiti putanje signala, kao i lako nadograđivati. Primer je poređenje dva neoznačena dvobitna binarna broja

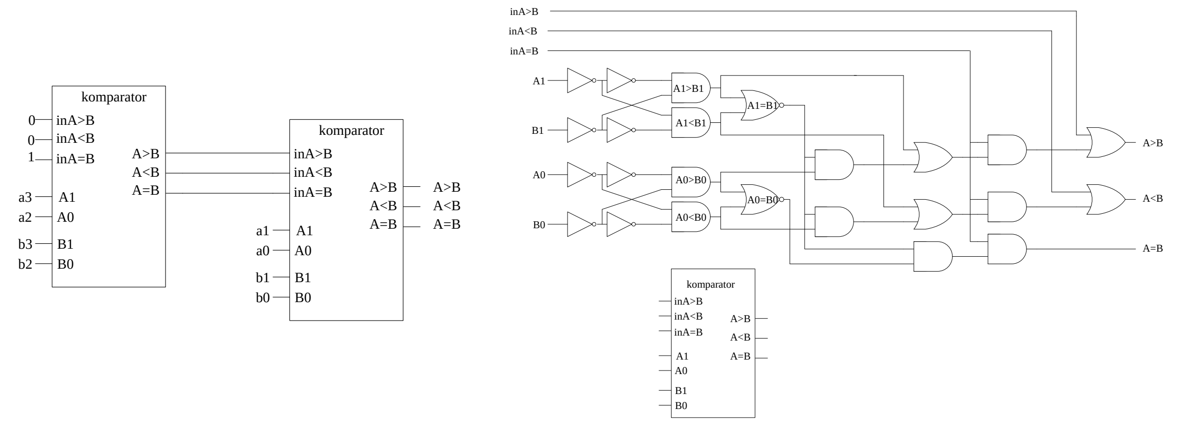




Komponenta se lako može proširiti i realizovati sa 4 bita sledeći logiku koja je prikazana. Samu logiku možemo da posmatramo „odozgo na dole“ u smislu: Ako je $A1 > B1$ onda je $A > B$ i ne interesuju nas niži biti, odnosno ako je $A1 < B1$ onda je $A < B$ i ne interesuju nas niži biti, odnosno samo ako je $A1 = B1$ videćemo na nižim bitima šta se dešava. Ovo možemo da proširimo u opštem slučaju. Poredimo bite Ai, Bi samo ako su svi biti više težine bili jednaki, a u tom slučaju ako je $Ai > Bi$ onda je $A > B$ i ne interesuju nas niži biti, odnosno ako je $Ai < Bi$ onda je $A < B$ i ne interesuju nas niži biti, odnosno samo ako je $Ai = Bi$ videćemo na nižim bitima šta se dešava.

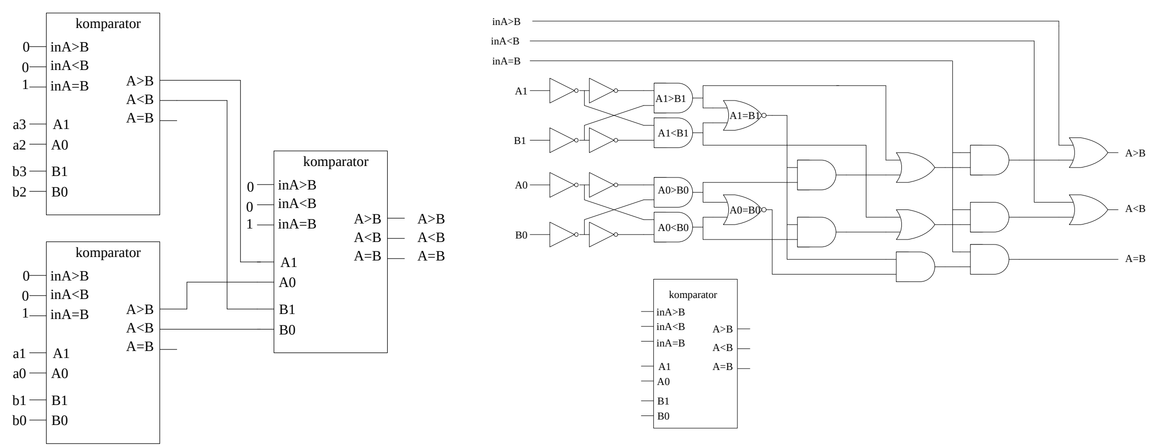


Mreže većih kapaciteta



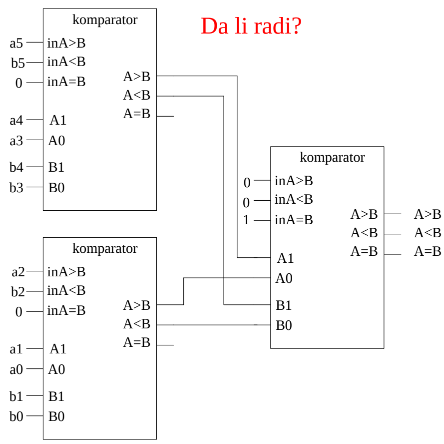
5

Mreže većih kapaciteta

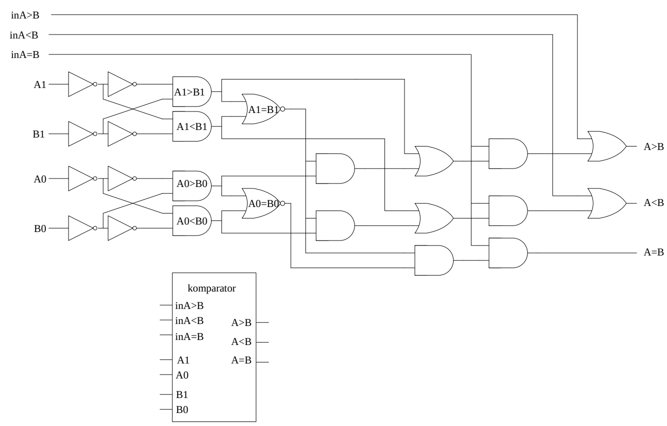


6

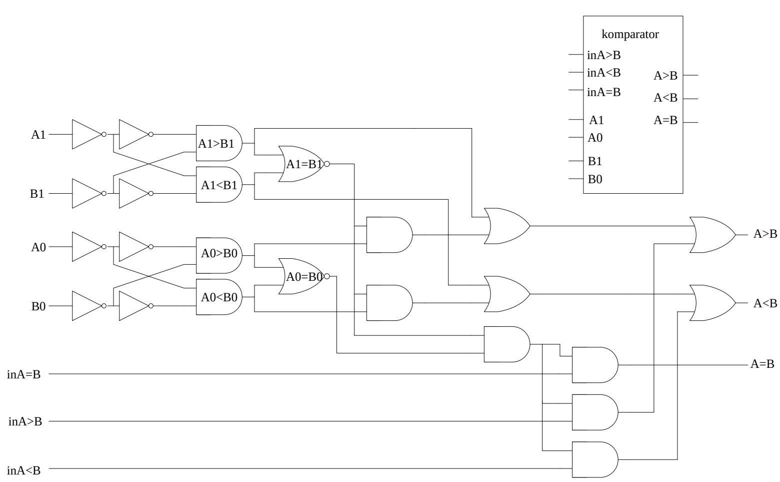
Mreže većih kapaciteta



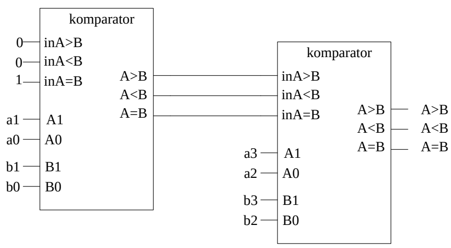
Da li radi?

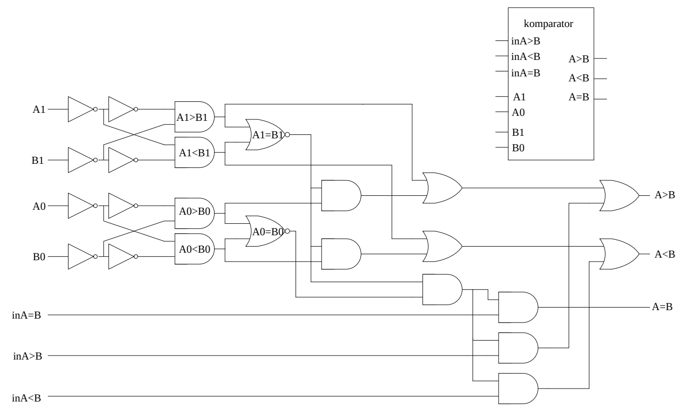


Samu logiku možemo da posmatramo i „odozdo na gore“ u smislu: Ako je $A0 > B0$ onda će rezultat zavisiti od viših bita, odnosno samo ako je $A1 = B1$ „tek onda“ je $A > B$. Itd. Ovo možemo da proširimo u opštem slučaju. Poredimo bite Ai, Bi i dotadašnji rezultat prosleđujemo na gore.



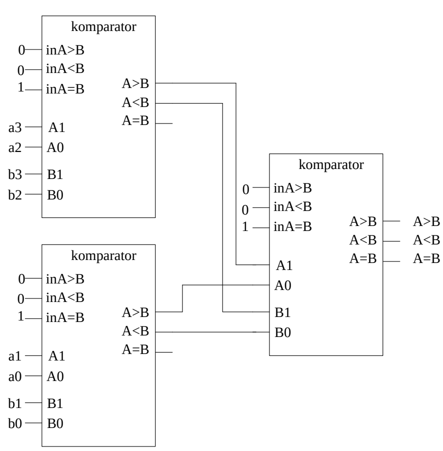
Mreže većih kapaciteta

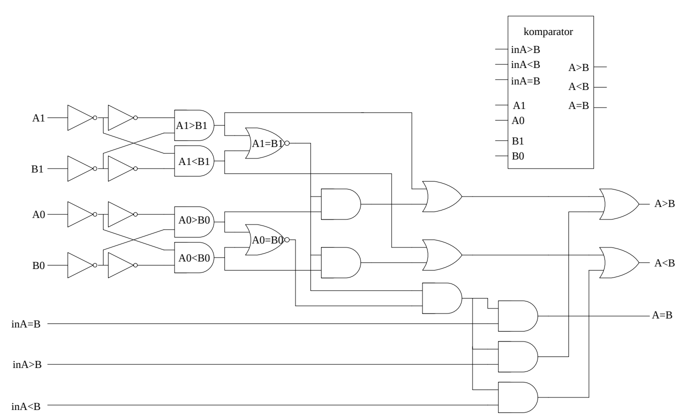




9

Mreže većih kapaciteta

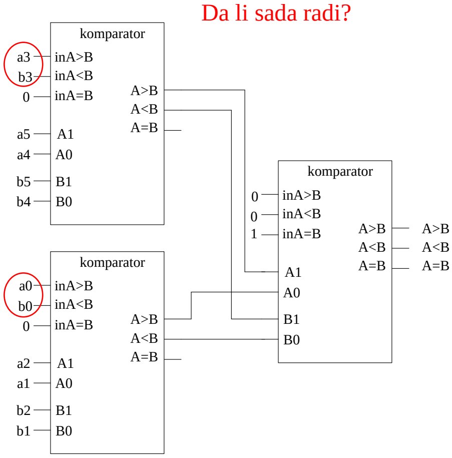


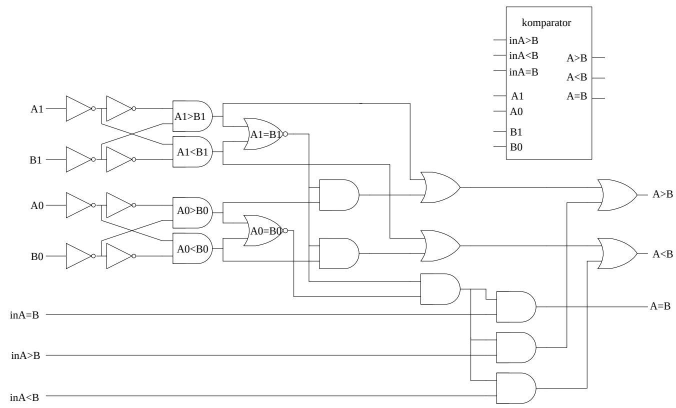


10

Mreže većih kapaciteta

Da li sada radi?





SABIRANJE

PRIMER: $326.23_{10} + 95.9_{10} = ?_{10}$

$$\begin{array}{r} 326.23 \\ +\ 95.9 \end{array} \quad \Rightarrow \quad \begin{array}{r} 326.23 \\ +\ 095.90 \end{array}$$

USAGLAŠENO MESTO TAČAKA

PODRAZUMEVANE VODEĆE I PRATEĆE NULE

$$\begin{array}{r} 326.23 \\ +\ 095.90 \\ \hline 422.13 \end{array}$$

Krećemo sabiranje od cifara najmanje težine

1. 3+0=3
2. 2+9=11 pišem 1 pamtim 1; rezultat 1 i 1 je prenos za sabiranje cifara naredne veće težine
3. …

K. Tačku stavljamo na isto mesto koje je bilo u operandima

Pretpostavke:

1. radimo sa fiksnom tačkom i ignorišemo njenu poziciju,
   smatramo da su operandi usaglašeni sa pozicijama tačke pa radimo sa celobrojnim vrednostima
2. radimo sa ograničenim brojem cifara *n* i za operande i za rezultate
3. imamo u registru potrebne vodeće i prateće nule

$a_i, b_i, s_i \in \{0,1,2, \dots r-2, r-1\}$

C – Carry, prenos 0 ili 1 bez obzira na osnovu brojnog sistema

$$\begin{array}{r} a_{n-1}a_{n-2} \dots a_i \dots a_1\ a_0 \\ b_{n-1}b_{n-2} \dots b_i \dots b_1\ b_0 \\ \hline s_n s_{n-1}s_{n-2} \dots s_i \dots s_1\ s_0 \end{array}$$

Sabiranje radimo u osnovi brojnog sistema

$$(a_i + b_i + C_{i-1})_{max} = r - 1 + r - 1 + 1 = 2r - 1 = r + r - 1$$

$C_i = 1$ $\quad s_i = r - 1$

$$s_n = 0 + 0 + C_{n-1} = 0 \ ili\ 1$$

$s_n = 1$ Prekoračenje opsega, rezultat nije moguće smestiti u *n* cifara

13

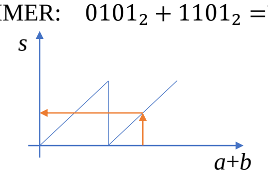
SIGNAL ZA PREKORAČENJE OPSEGA
C posle sabiranja cifara najveće težine
Izlazni keri bit jednak jedinici

PRIMER: $326.23_7 + 45.4_7 = ?_7$

C	001000
	32623
	+ 04540
	40463

404.63 nema prekoračenja opsega

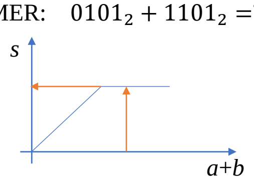
PRIMER: $0101_2 + 1101_2 = ?_2$



C	11010
	0101
	+ 1101
	10010

0010 ima prekoračenja opsega

PRIMER: $0101_2 + 1101_2 = ?_2$



Sa zasićenjem

C	11010
	0101
	+ 1101
	10010 → 1111

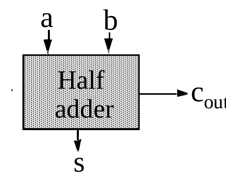
ima prekoračenja opsega, zasićenje

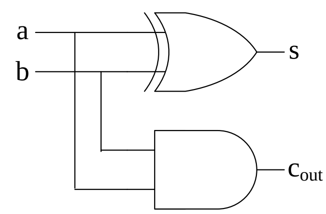
14

Zašto sve ovo radimo?

Polusabirač

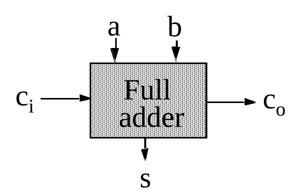
a b	c_{out}	s
0 0	0	0
0 1	0	1
1 0	0	1
1 1	1	0





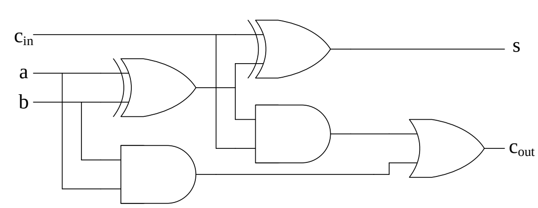
Potpuni sabirač

c_{in} a b	c_{out}	s
0 0 0	0	0
0 0 1	0	1
0 1 0	0	1
0 1 1	1	0
1 0 0	0	1
1 0 1	1	0
1 1 0	1	0
1 1 1	1	1



$$s = c_{in} \oplus a \oplus b$$

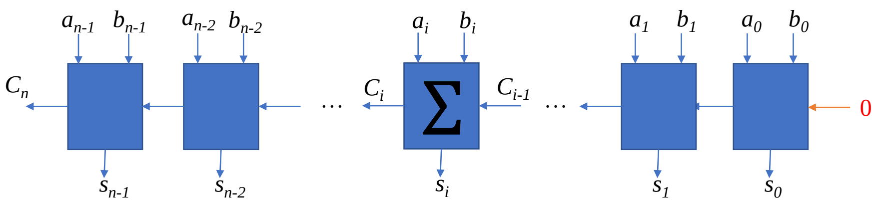
$$c_{out} = ab + c_{in}(a \oplus b)$$



$$r = c_{in}ab + \overline{c_{out}}(c_i + a + b)$$

Zašto sve ovo radimo?

Digitalni sistem – binarno sabiranje – sabirač – ideja iz algoritamskog sabiranja



Worst case kašnjenje

Kašnjenje za najgori slučaj

$$t_{adder} = (N-1)t_{carry} + t_{sum}$$

Raste linearno sa brojem bita

$$t_{adder} = O(N)$$

Cilj: Napraviti što brži prenos carry bita

Oduzimanje

PRIMER: $326.23_{10} - 95.9_{10} = ?_{10}$

```
  326.23            326.23
-  95.9          - 095.90
```

USAGLAŠENO MESTO TAČAKA

PODRAZUMEVANE VODEĆE I PRATEĆE NULE

```
  326.23
- 095.90
--------
  230.33
```

Krećemo oduzimanje od cifara najmanje težine

1. 3-0=3
2. 2-9=? „Ne može“ pa ćemo pozajmiti jedinicu od cifara veće težine.
   12-9=3 i pozajmica 1

3. 6-(5+1) ili (6-1)-5 kako smo već učili, pošto imamo pozajmicu jedan zbog oduzimanja cifara manje težine

…

K. Tačku stavljamo na isto mesto koje je bilo u operandima

Pretpostavke:

1. radimo sa fiksnom tačkom i ignorišemo njenu poziciju,
   smatramo da su operandi usaglašeni sa pozicijama tačke pa radimo sa celobrojnim vrednostima
2. radimo sa ograničenim brojem cifara *n* i za operande i za rezultate
3. imamo u registru potrebne vodeće i prateće nule

$a_i, b_i, s_i \in \{0,1,2, \dots r-2, r-1\}$

B – Borrow, pozajmica 0 ili 1 bez obzira na osnovu brojnog sistema

$$\begin{array}{r} a_{n-1}a_{n-2} \dots a_i \dots a_1\ a_0 \\ b_{n-1}b_{n-2} \dots b_i \dots b_1\ b_0 \\ \hline s_n s_{n-1} s_{n-2} \dots s_i \dots s_1\ s_0 \end{array}$$

Oduzimanje radimo u osnovi brojnog sistema

$$(a_i - B_{i-1} - b_i)_{min} = 0 - 1 - (r-1) = -r + 0$$

$B_i = 1$ $\quad s_i = 0$

$s_n = 0 - 0 - B_{n-1} = 0\ ili\ -1$???

$s_n = -1$ Prekoračenje opsega, rezultat nije moguće smestiti u *n* cifara???

Javlja se kada je *a* < *b*
odnosno kada rezultat treba da bude negativan
O ovome kasnije

PRIMER: $326.23_7 - 45.4_7 = ?_7$

B	010100
	32623
-	04540
	25053

250.53

PRIMER: $1100_2 - 0101 = ?_2$

B	01110
	1100
-	0101
	0111

0111

PRIMER: $0101_2 - 1100_2 = ?_2$

B	…1110000
	0101
-	1100
	…1111001

…1111001 → 1001 Rezultat treba da je -7 a vrednost je 9
Međutim ako rezultat posmatramo kao negativan broj u drugom komplementu onda jeste -7

Papir i olovka

a-b=?

Ako je a=b rezultat =0

Ako je a>b klasično oduzimanje

Ako je a<b rezultat =-(b-a) radimo klasično oduzimanje b-a i rezultatu menjamo znak

Mogli bi ovu proceduru da sprovedemo i u realizaciji digitalnog sistema pogotovo na primer ako su nam negativni brojevi znak i apsoluta vrednost.
Treba nam komparator ali je njega lako napraviti u binarnom brojnom sistemu.
Napravili smo ga.

Kada smo radili sabiranje radili smo samo sa pozitivnim vrednostima međutim ako imamo i negativne brojeve nama trebaju sabiranja

$$a, b \geq 0$$
$$a + b$$
$$(-a) + b$$
$$a + (-b)$$
$$(-a) + (-b)$$

Kada razmatramo oduzimanje radili smo samo sa pozitivnim vrednostima međutim ako imamo i negativne brojeve nama trebaju oduzimanja

$$a, b \geq 0$$
$$a - b$$
$$(-a) - b$$
$$a - (-b)$$
$$(-a) - (-b)$$

Uočiti da sabiranje i oduzimanje možemo da posmatramo kako smo naučili, Papir i olovka

$$a, b \geq 0$$

$$a - (-b) = a + b$$
$$(-a) - b = (-a) + (-b) = -(a + b)$$

$$a - b = a + (-b)$$
$$(-a) - (-b) = (-\mathrm{a}) + \mathrm{b}$$

Da li nam oduzimanje kao osnovna operacija treba? Oduzimač?
Da se podsetimo šta smo rekli ako je $a<b$

$$a - b = -(b - a)$$

U digitalnom sistemu sa drugim komplementom i n cifara pošto znamo da će rezultat biti negativan

$$a - b = -(b - a) \equiv 2^n - (b - a) = a + (2^n - b) \equiv a + (-b)$$

Oduzimanje smo uradili sabiranjem sa negativnom vrednosti

Da li smo ovo mogli da uradimo ne razmatrajući odnose a i b

$$a, b \geq 0$$

$$a - b \equiv a + (2^n - b) = 2^n - (b - a) = 2^n + (a - b)$$

$a < b$
$a + (2^n - b) = 2^n - (b - a)$ Dobijamo negativan rezultat prikazan u drugom komplementu, prilikom sabiranja neće se pojaviti izlazni keri

$a > b$
$a + (2^n - b) = 2^n + (a - b)$ Dobijamo pozitivan rezultat i prilikom sabiranja će se pojaviti izlazni keri

PRIMER: $00101_2 - 01100_2 = ?_2$ Primer koji smo imali ali sa dodatim 5. bitom zbog znaka

$$00101 - 01100 = 00101 + (100000 - 01100) = 00101 + 10100$$

```
C  001000
    00101
  + 10100
  -------
    11001  →  5-12=-7
```

11001
00110
00111

23

PRIMER: $0101_2 - 1100_2 = ?_2$

Uočite da smo na kraju posle direktnog oduzimanja dodali 100000
„zaustavili bi proces“ tj dobijali „vodeće nule“.
Zato sam i rekao da je komplement osnove „prirodan“ za prikazivanje negativnih vrednosti

```
B   ...111 0000
           0101
         - 1100
    -----------
     ...111 1001
         +1 0000
    -----------
     ...000 1001
```

PRIMER: $45_{10} - 67_{10} = ?_{10}$

```
B   ...111 10
           45
         - 67
    ---------
     ...999 78
        +1 00
    ---------
     ...000 78
```

100-78=22
-(100-78)= -22
45-67= -22

24

Oduzimanje uvek radimo sabiranjem sa komplementiranom vrednosti

$$a - b \equiv a + (2^n - b)$$

bez obzira da li su *a* i/ili *b* pozitivni ili negativni

Digresija

$$b < 0$$
$$a - b = a + |b|$$
$$b \equiv (2^n - |b|)$$
$$-b \equiv 2^n - (2^n - |b|) = |b|$$

Na višem nivou digitalnog sistema postojaće operacije ADD i SUB
ali se operacija SUB izvodi na prethodno opisani način.
Nećemo praviti digitalni oduzimač.

Da li se uvek rezultati ADD i SUB operacija mogu smestiti u *n* bitni rezultat. Kada nastupa prekoračenje.

$$a, b \geq 0$$

$$a + (-b) \equiv a + (2^n - b) = 2^n - (b - a) = 2^n + (a - b)$$
$$(-a) + b \equiv (2^n - a) + b = 2^n - (a - b) = 2^n + (b - a)$$

$$a + b \equiv a + b$$
$$(-a) + (-b) \equiv (2^n - a) + (2^n - b) = 2^n + 2^n - (a + b)$$

1. i 2. slučaj: Ako su operandi različitog znaka $a_{n-1} \neq b_{n-1}$ nema prekoračenja, ignoriše se eventualna pojava izlaznog kerija

3. i 4. slučaj: Ako su operandi istog znaka $a_{n-1} = b_{n-1}$
nema prekoračenja ako je i rezultat istog znaka, ignoriše se eventualna pojava izlaznog kerija

Za *n* bitni registar gde je *n*-1 cifara + bit znaka

$a_{max} = b_{max} = 2^{n-1} - 1$	$a_{max} + b_{max} = 2^n - 2$	$c_{n-1} = 1$
$\lvert a_{min}\rvert = \lvert b_{min}\rvert = 2^{n-1}$	$2^n + 2^n - (\lvert a_{min}\rvert + \lvert b_{min}\rvert) = 2^n$	$c_{n-1} = 0$

Negativni brojevi u prvom komplementu.

$$a, b \geq 0$$

$$a + (-b) \equiv a + (2^n - 1 - b) = 2^n - 1 - (b - a) = 2^n - 1 + (a - b)$$
$$(-a) + b \equiv (2^n - 1 - a) + b = 2^n - 1 - (a - b) = 2^n - 1 + (b - a)$$

$$a + b \equiv a + b$$
$$(-a) + (-b) \equiv (2^n - 1 - a) + (2^n - 1 - b) = 2^n - 1 + 2^n - 1 - (a + b)$$

1. i 2. slučaj: Ako su operandi različitog znaka $a_{n-1} \neq b_{n-1}$ nema prekoračenja,

3. i 4. slučaj: Ako su operandi istog znaka $a_{n-1} = b_{n-1}$
nema prekoračenja ako je i rezultat istog znaka

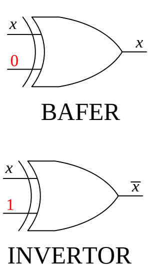
1. i 2. slučaj: Ako je rezultat pozitivan pojaviće se keri bit (plavo) ali treba dodati još 1 (crveno) da bi rezultat bio korektan.

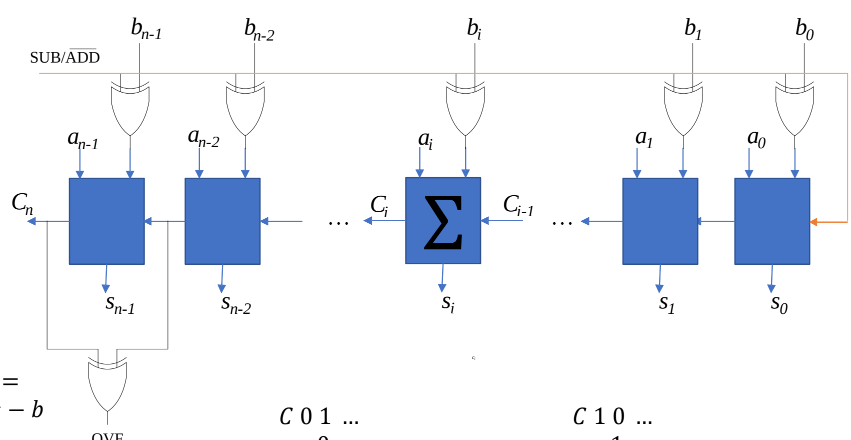
4. slučaj: Pojaviće se keri bit (plavo) ali treba dodati još 1 (crveno) da bi rezultat bio korektan.

U opštem slučaju na rezultat treba dodati keri bit.

27

Oduzimač/sabirač u drugom komplementu $a \mp b$





$$SUB \equiv a - b \equiv a + \bar{b} + 1 =$$
$$a + 2^n - 1 - b + 1 = a + 2^n - b$$

$$ADD \equiv a + b$$

OVF

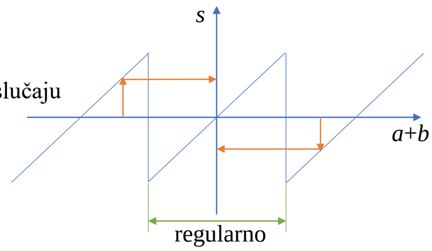
C 0 1 ...	C 1 0 ...
0 a_{n-2} ... $a_i \ldots a_1\ a_0$	1 a_{n-2} ... $a_i \ldots a_1\ a_0$
0 b_{n-2} ... $b_i \ldots b_1\ b_0$	1 b_{n-2} ... $b_i \ldots b_1\ b_0$
1 s_{n-2} ... $s_i \ldots s_1\ s_0$	0 s_{n-2} ... $s_i \ldots s_1\ s_0$

Za prvi komplement NE VAŽI

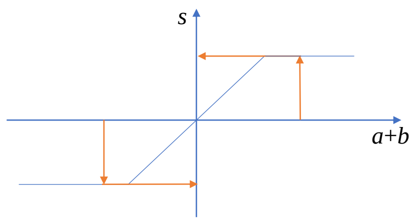
28

Prekoračenje opsega –zašto nam treba informacija?

Zbir dve negativne vrednosti u slučaju OVF daje pozitivnu vrednost



Zbir dve pozitivne vrednosti u slučaju OVF daje negativnu vrednost



Česta realizacija u digitalnoj obradi signala

U slučaju OVF zasićenje

29

Proširenje opsega brojeva – ekstenzija znaka

D_3	D_2	D_1	D_0
b_3	b_2	b_1	b_0

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0

NEOZNAČENI BROJEVI

D_3	D_2	D_1	D_0
0	1	1	0

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	0	0	0	0	1	1	0

D_3	D_2	D_1	D_0
1	1	1	0

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	0	0	0	1	1	1	0

30

Proširenje opsega brojeva – ekstenzija znaka

OZNAČENI BROJEVI

KOMPLEMENT

D_3	D_2	D_1	D_0
0	1	1	0

→

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	0	0	0	0	1	1	0

D_3	D_2	D_1	D_0
1	1	1	0

→

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	1	1	1	1	1	1	0

$$2^8 - a = 2^8 - 2^4 + 2^4 - a = 2^4(2^4 - 1) + 2^4 - a$$

1111

Pomeranje u levo za 4 mesta

ZNAK I APSOLUTNA VREDNOST

D_3	D_2	D_1	D_0
1	1	1	0

→

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	0	0	1	1	0

31

LOGIČKO I ARITMETIČKO POMERANJE

LSL logical shift left

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0

→

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
b_6	b_5	b_4	b_3	b_2	b_1	b_0	0

LSR logical shift right

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0

→

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	b_7	b_6	b_5	b_4	b_3	b_2	b_1

ASL arithmetic shift left

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0

→

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
b_6	b_5	b_4	b_3	b_2	b_1	b_0	0

ASR arithmetic shift right

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0

→

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
b_7	b_7	b_6	b_5	b_4	b_3	b_2	b_1

Shift left = puta 2
Shift right = podeljeno sa 2

Kada nastupa OVF?

32

IDEJE

INC-increment, +1

	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
	b_7	b_6	b_5	b_4	0	1	1	1
+	0	0	0	0	0	0	0	1
	b_7	b_6	b_5	b_4	1	0	0	0

DEC-decrement, -1

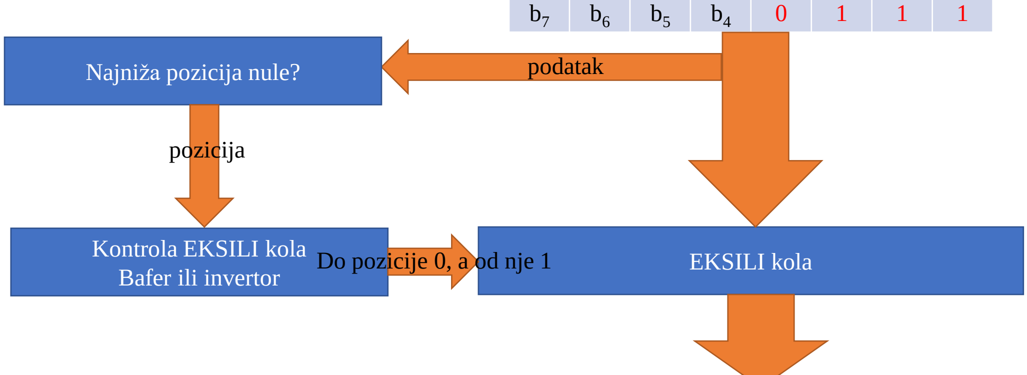
	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
	b_7	b_6	b_5	1	0	0	0	0
-	0	0	0	0	0	0	0	1
	b_7	b_6	b_5	0	1	1	1	1

33

IDEJA – da li nam treba sabirač

INC-increment, +1

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
b_7	b_6	b_5	b_4	0	1	1	1

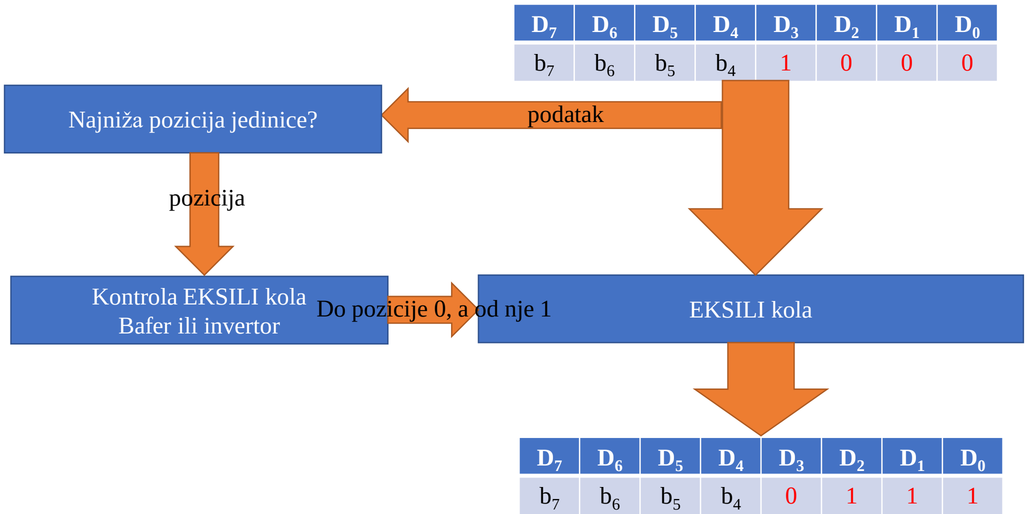


D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
b_7	b_6	b_5	b_4	1	0	0	0

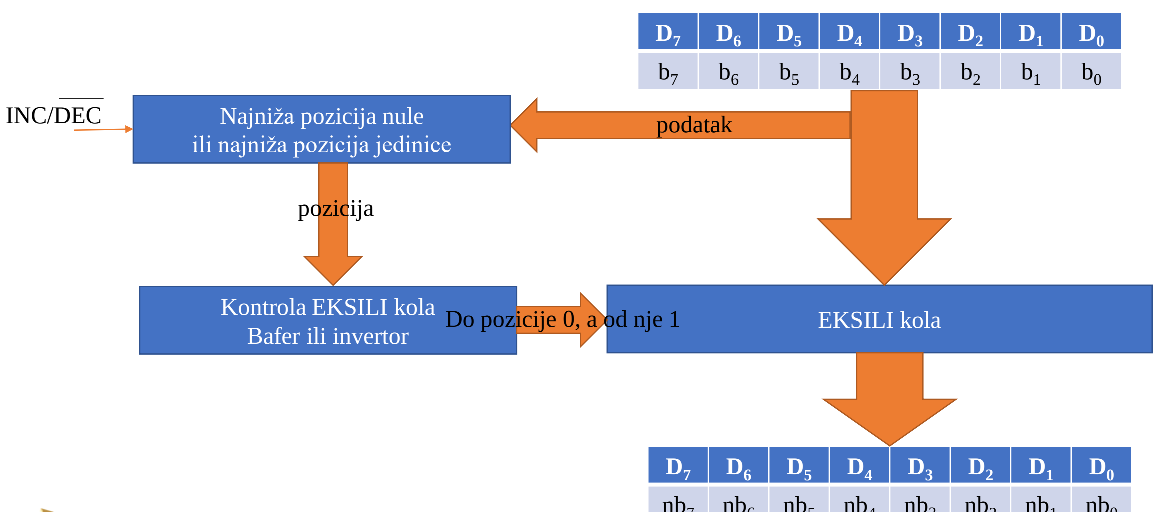
34

IDEJA – da li nam treba oduzimač

DEC-decrement, -1



IDEJA – da li nam treba sabirač/oduzimač



Množenje

NEOZNAČENO 5x3=101x011=15=1111

			1	0	1	x	0	1	1
			1	0	1				
+		1	0	1					
+	0	0	0						
	0	1	1	1	1				

Parcijalni proizvodi

logička I operacija bita operanada

				1	0	1	x	0	1	1
	0	0	0	1	0	1				
+	0	0	1	0	1	0				
+	0	0	0	0	0	0				
	0	0	1	1	1	1				

0 – ASL, x2

0 - Proširenje opsega

$$(2^n-1)(2^n-1) = (2^{2n}-1) - (2^{n+1}-1) + 1$$

Treba nam $2n$ mesta za smeštanje rezultata

37

Množenje

OZNAČENO

Trobitni operandi, drugi komplement

(-3)x3=-9
101x011=110111

				1	0	1	x	0	1	1
	0	0	0	1	0	1				
+	0	0	1	0	1	0				
+	0	0	0	0	0	0				
	0	0	1	1	1	1				

?

38

Množenje

OZNAČENO

Trobitni operandi, drugi komplement

(-3)x3=-9
101x011=110111

				1	0	1	x	0	1	1
	1	1	1	1	0	1				
+	1	1	1	0	1	0				
+	0	0	0	0	0	0				
1	1	1	0	1	1	1				

OZNAČENO - EKSTENZIJA ZNAKA

Množenje

OZNAČENO

Trobitni operandi, drugi komplement

3x (-3) =-9
011x101=110111

				0	1	1	x	1	0	1
	0	0	0	0	1	1				
+	0	0	0	0	0	0				
+	0	0	1	1	0	0				
	0	0	1	1	1	1				

? — uradili smo dobru ekstenziju znaka

Množenje

OZNAČENO

Trobitni operandi, drugi komplement

3x (-3) =-9

011x101=110111

				0	1	1	x	1	0	1
	0	0	0	0	1	1				
+	0	0	0	0	0	0				
-	0	0	1	1	0	0				
	1	1	0	1	1	1				

				0	1	1	x	1	0	1
	0	0	0	0	1	1				
+	0	0	0	0	0	0				
+	1	1	0	1	0	0				
	1	1	0	1	1	1				

Bit najveće težine u množiocu je 1,
što pokazuje oduzimanje, negativnu vrednost,
parcijalni proizvod treba da bude negativan

Množenje

OZNAČENO

Trobitni operandi, drugi komplement

(-3) x (-3) =+9

101x101=001001

				1	0	1	x	1	0	1
	1	1	1	1	0	1				
+	0	0	0	0	0	0				
+	0	0	1	1	0	0				
1	0	0	1	0	0	1				

Množenje

Dva različita algoritma za označeno i neoznačeno množenje
Po pravilu digitalni sistemi, procesori, imaju dve operecije množenja

MPY - neoznačeno
MPYS – signed - označeno

Označeno množenje je bilo moguće uraditi i na sledeći način

- Zapamtiti znak
- Učiniti sve pozitivnim
- Uraditi množenje
- Definisati znak rezultata

```
if sign(a)!=sign(b) then s = true, else s=false
    a = abs(a)
    b = abs(b)
    p = a*b
    negate p if s = true
```

Za označeno množenje često se koristi

Booth's Algorithm

Podsećanje

$$D = D_{n-1}D_{n-2}\dots D_1D_0 = D_{n-1}2^{n-1} + D^*$$
$$D_{n-1} = 0 \quad \Rightarrow \quad D_t = D^*$$
$$D_{n-1} = 1 \quad \Rightarrow \quad D_t = -(2^n - D) = -(2^n - D_{n-1}2^{n-1} - D^*) = -D_{n-1}2^{n-1} + D^*$$

Za proizvod b*a formirajmo parcijalne proizvode na sledeći način

$a=(a_{31}a_{30}a_{29}a_{28} \dots a_3a_2a_1a_0)_2$

$a_{-1}=0$

$(a_{-1} - a_0) \times b \times 2^0$
$(a_0 - a_1) \times b \times 2^1$
$(a_1 - a_2) \times b \times 2^2$
…
$(a_{29} - a_{30}) \times b \times 2^{30}$
$(a_{30} - a_{31}) \times b \times 2^{31}$

i saberimo

$= b \times (- a_{31}\, 2^{31} + a_{30}\, 2^{30} + \dots + a_1\, 2^1 + a_0\, 2^0)$

$a=a_{31}a_{30}a_{29}a_{28} \dots a_3a_2a_1a_0$ $a_{-1}=0$

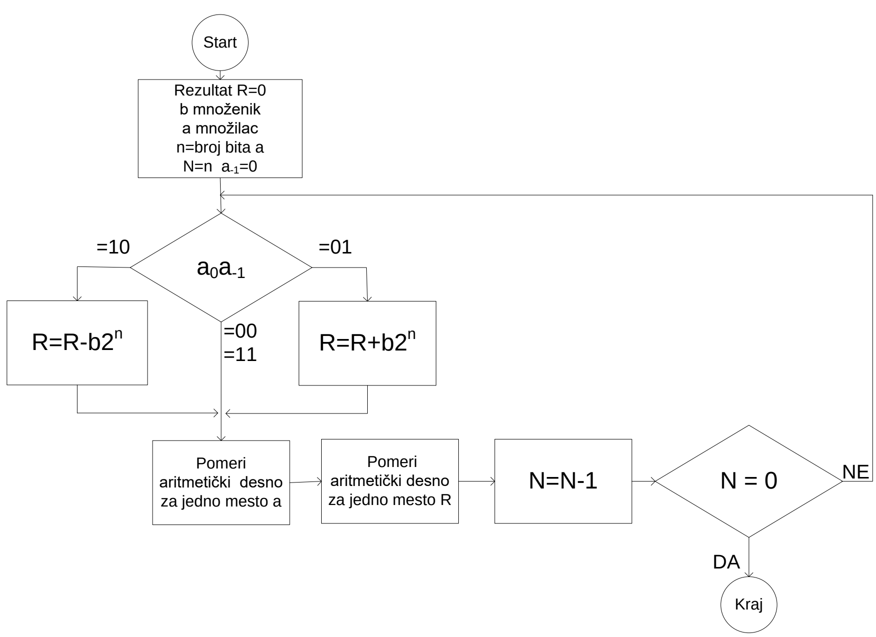
$(a_{-1} - a_0) \times b \times 2^0$
$(a_0 - a_1) \times b \times 2^1$
$(a_1 - a_2) \times b \times 2^2$
…
$(a_{29} - a_{30}) \times b \times 2^{30}$
$(a_{30} - a_{31}) \times b \times 2^{31}$

$=b \times (- a_{31}\, 2^{31} + a_{30}\, 2^{30} + \dots + a_1\, 2^1 + a_0\, 2^0)$

Ako posmatramo dva susedna bita u operandu a

- **00: 0-0 = nop**
- **01: 1-0 = add**
- **10: 0-1 = sub**
- **11: 1-1 = nop**

Iteracija	Operand **b**=0010 **a**=0110	Korak	Parcijalni proizvod 0000 0000
N=4	0010 01100	00: no op arith>> 1	0000 0000 0000 0000
N=3	0010 00110	10: $R=R-b2^n$ arith>> 1	1110 0000 1111 0000
N=2	0010 00011	11: no op arith>> 1	1111 0000 1111 1000
N=1	0010 00001	01: $R=R+b2^n$ arith>> 1	0001 1000 0000 1100
N=0		kraj	0000 1100



ODSECANJE REZULTATA MNOŽENJA

$$a_{n-1}a_{n-2} \ldots a_1\, a_0 \times b_{n-1}b_{n-2} \ldots b_1\, b_0 = c_{2n-1}c_{2n-2} \ldots c_n c_{n-1}c_{n-2} \ldots c_1\, c_0$$

Ako rezultat množenja smeštamo takođe u n bitni registar kod celobrojnog množenja mora da se vodi računa da ne dođe do prekoračenja opsega, odnosno da rezultat može da stane u n bitni registar

$c_{2n-1}c_{2n-2} \ldots c_n c_{n-1} = 00 \ldots 00$	Za pozitivan rezultat, označeno množenje
$c_{2n-1}c_{2n-2} \ldots c_n c_{n-1} = 11 \ldots 11$	Za negativan rezultat, označeno množenje
$c_{2n-1}c_{2n-2} \ldots c_n = 00 \ldots 00$	Neoznačeno množenje

Greška koju bi napravili eventualnim prekoračenjem opsega bila bi „ogromna“.
Videli smo to kod sabiranja.

ODSECANJE REZULTATA MNOŽENJA

U sistemima gde su prekoračenja moguća, odnosno operandi nisu unapred „poznati“, koristi se fiksna tačka iza bita najveće težine. Puno se koristi u digitalnoj obradi signala.

$$a_{n-1}.a_{n-2} \ldots a_1\, a_0 \times b_{n-1}.b_{n-2} \ldots b_1\, b_0 = c_{2n-1}.c_{2n-2} \ldots c_n c_{n-1}c_{n-2} \ldots c_1\, c_0$$

Prilikom množenja ne može da dođe do prekoračenja opsega

$$-1 \le a < 1, -1 \le b < 1 \implies -1 < c < 1$$

uz izuzetak $a = -1,\ b = -1$ što je moguće kontrolisati.

Biti $c_{2n-1}.c_{2n-2} \ldots c_n$ se smeštaju u n bitni registar i koriste za dalju obradu

Biti $c_{n-1}c_{n-2} \ldots c_1\, c_0$ se odbacuju, odseca se rezultat množenja

Primer sa četvorobitnim registrima i notacijama indeksa iz brojnih sistema

$$Q_t\{c_0.c_{-1}c_{-2}c_{-3}c_{-4}c_{-5}c_{-6}c_{-7}\} \rightarrow c_0.c_{-1}c_{-2}c_{-3}0000 \rightarrow c_0.c_{-1}c_{-2}c_{-3}$$

bez obzira da li je broj negativan ili pozitivan.
Uočiti
$c_0.c_{-1}c_{-2}c_{-3} = c_0.c_{-1}c_{-2}c_{-3}0000 = c_0.c_{-1}c_{-2}c_{-3}c_{-4}c_{-5}c_{-6}c_{-7} - 0.000c_{-4}c_{-5}c_{-6}c_{-7}$

Najveća apsolutna greška je u ovom primeru je $0.0001111 = \frac{15}{2^7}$

U opštem slučaju $c_{n-1}c_{n-2} \ldots c_1\ c_0 = 11 \ldots 11 \quad \varepsilon = \frac{2^n - 1}{2^{2n-1}}$

Odsecanjem se rezultat množenja smanjuje.
Uočiti da se po apsolutnoj vrednosti pozitivni brojevi smanjuju
a negativni po apsolutnoj vrednosti rastu

ZAOKRUŽIVANJE REZULTATA MNOŽENJA

Da bi se smanjila greška odsecanja u mnogim sistemima se radi zaokruživanje rezultata.

$$a_{n-1}.a_{n-2} \ldots a_1\ a_0 \times b_{n-1}.b_{n-2} \ldots b_1\ b_0 = c_{2n-1}.c_{2n-2} \ldots c_n c_{n-1} c_{n-2} \ldots c_1\ c_0$$

Vrednost $c_{2n-1}.c_{2n-2} \ldots c_n + \mathrm{r}$ se smešta u *n* bitni registar i koriste za dalju obradu
r je dodatna vrednost zaokruživanja i zavisi od načina koji se primenjuje.
Sigurno je manja ili jednaka od vrednosti težine najnižeg bita $c_n \equiv c_{-n+1}$

Zaokruživanje na dole – prema minus beskonačno – praktično odsecanje

Zaokruživanje na gore – prema plus beskonačno – $r = \frac{1}{2^{n-1}}$ dodaje se vrednost bita najmanje težine

Zaokruživanje prema nuli

Zaokruživanje od nule

Zaokruživanje najbližoj vrednosti $r = c_{n-1}$ pa onda prema parnoj, neparnoj itd…

DELJENJE

Primer 74:8=1001010:1000

	1	0	0	1	0	1	0	:	1	0	0	0	=	1	0	0	1
-	1	0	0	0													
M, R>=0	0	0	0	1													
		0	0	1	0												
>>, -		1	0	0	0												
N, R=<0																	
			0	1	0	1											
>>, -			1	0	0	0											
N, R=<0																	
				1	0	1	0										
>>, -				1	0	0	0										
M, R>=0				0	0	1	0										

53

DELJENJE

Primer 74:8=1001010:1000

	1	0	0	1	0	1	0	:	1	0	0	0	=	1	0	0	1
-	1	0	0	0	0	0	0										
R>=0	0	0	0	1	0	1	0										
>>, -		1	0	0	0	0	0										
R=<0		-	-	-	-	-	-										
+	0	0	0	1	0	1	0										
>>, -			1	0	0	0	0										
R=<0			-	-	-	-	-										
+	0	0	0	1	0	1	0										
>>, -				1	0	0	0										
R>=0				0	0	1	0										

Rezultat je 1 ako Delilac <= Deljenik, inače 0
- Kako ALU zna da li je ovo tačno?
- Oduzmi i ako je “rezultat” manji od nule rezultat je nula
- Pomeri i probaj ponovo

54

Označeno deljenje

- Zapamtiti znak
- Učiniti sve pozitivnim
- Uraditi deljenje
- Definisati znak rezultata

```
if sign(a)!=sign(b) then s = true, else s=false
    a = abs(a)
    b = abs(b)
    p = a/b
    negate p if s = true
```