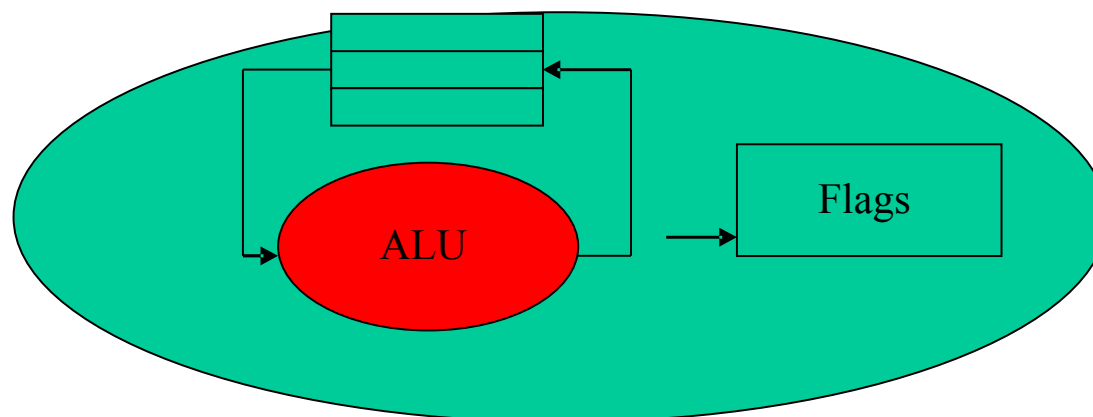


Základné aritmeticko-logické operácie

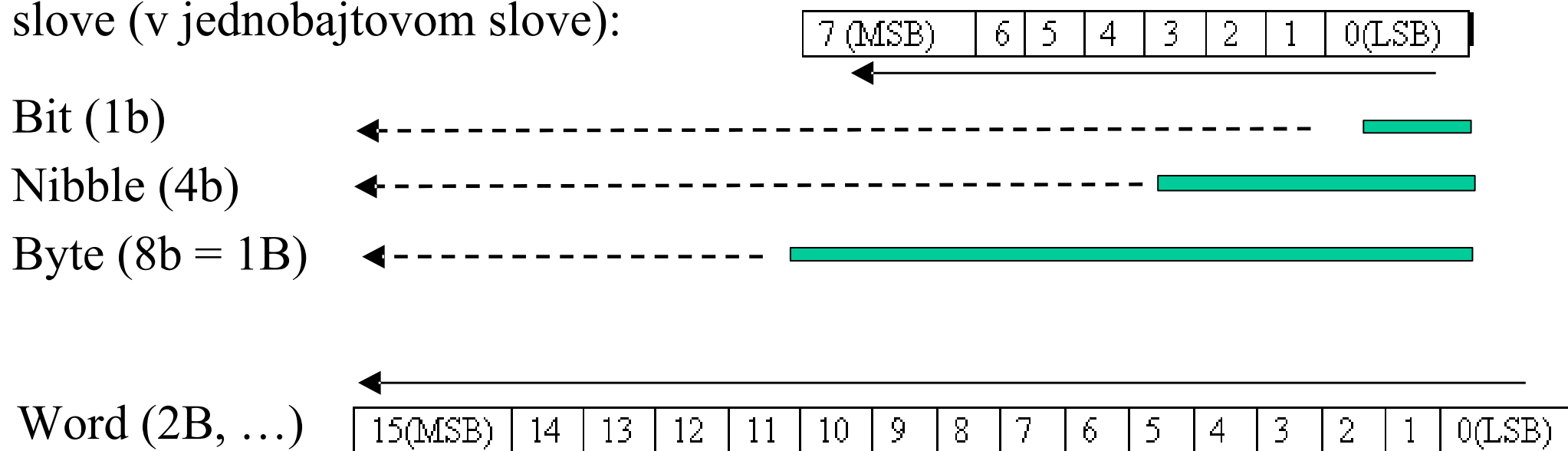
Aritmeticko-logické operácie sa väčšinou vykonávajú v aritmeticko-logickej jednotke (**ALJ**) (Arithmetics and Logic Unit (**ALU**)), ktorá je súčasťou centrálnej procesorovej jednotky (Central Process Unit **CPU**)
Resp. (Central Processing Unit) - hlavná vykonávacia jednotka



Logické operácie

Logické operácie sa uskutočňujú nad pamäťovými miestami obyčajne s registrami **CPU**, pričom sa operácia **vykonáva so všetkými dvojicami bitov pamäťových miest**. Pamäťové miesta (slová) môžu byť jedno a viacbajtové.

Bity v slovách označujeme indexom, ktorého počiatočná hodnota je 0, a index narastá sprava doľava. Označovanie bitov v 8-bitovom slove (v jednobajtovom slove):



LSB – **L**east **S**ignificant **B**it (bit s najmenšou váhou)

MSB – **M**ost **S**ignificant **B**it (bit s najväčšou váhou)

Pravidlá základných logických operácií pre jednobitové operandy:

negácia (NOT)

$$B = \text{not}(A); B = \overline{A}$$

A	B
0	1
1	0

logický súčet (OR)

$$C = A \text{ or } B; C = A + B$$

A	B	C
0	0	0
1	0	1
0	1	1
1	1	1

logický súčin (AND)

$$C = A \text{ and } B; C = A . B$$

A	B	C
0	0	0
1	0	0
0	1	0
1	1	1

neekvivalencia (XOR- eXclusive OR)

$$C = A \text{ XOR } B; C = a \oplus b$$

A	B	C
0	0	0
1	0	1
0	1	1
1	1	0

Jedno alebo druhé, ale nie všetky naraz (oboje), 1 až n-1 z n. U nás je n=2
Niektoré realizácie:
Práve jedno.

Príklady viacbitových (8-bitové operandy - bajt) logických operácií

negácia (*unárna operácia*)

$$B = NOT(A) \quad [b = \sim a]$$

<i>A</i>	0	1	0	1	1	1	0	0
<i>B</i>	1	0	1	0	0	0	1	1

logický súčin (AND)

$$C = A \text{ AND } B \quad [c = a \& b]$$

<i>A</i>	0	1	0	1	1	1	1	0
<i>B</i>	1	0	0	1	0	1	1	1
<i>C</i>	0	0	0	1	0	1	1	0

logický súčet (*binárna operácia*)

$$C = A \text{ OR } B \quad [c = a | b]$$

<i>A</i>	0	1	0	1	1	1	1	0
<i>B</i>	1	0	0	1	0	1	1	1
<i>C</i>	1	1	0	1	1	1	1	1

neekvivalencia (XOR)

$$C = A \text{ XOR } B \quad [c = a \wedge b]$$

<i>A</i>	0	1	0	1	1	1	1	0
<i>B</i>	1	0	0	1	0	1	1	1
<i>C</i>	1	1	0	0	1	0	0	1

XOR sa tiež nazýva **exluzívny súčet, súčet modulo 2, EOR**.

V jazyku C treba odlišovať **&**, **|** od logických spojok **&&** a **||**.

Použitie logických operácií

negácia

zmena logickej hodnoty všetkých bitov slova **na opačnú logickú hodnotu** nezávisle od predchádzajúcej logickej hodnoty bitu (napr. kódovanie záporných čísel)

logický súčet

<i>A</i>	0	1	0	1	0	1	0	1
<i>B</i>	1	1	0	0	0	0	1	1
<i>C</i>	1	1	0	1	0	1	1	1

<i>A</i>	1	0	1	0	1	0	1	0
<i>B</i>	1	1	0	0	0	0	1	1
<i>C</i>	1	1	1	0	1	0	1	1

vnútenie logickej jednotky do ľubovoľnej pozície v slove tzv. maskou (**B**). Výsledok logického súčtu bude mať hodnotu **1** v bitoch, kde má hodnotu **1** operand **B**, v ostatných bitoch sa zachová pôvodná logická hodnota **X**

1 1 X X X X 1 1

logický súčin

<i>A</i>	0	1	0	1	0	1	0	1
<i>B</i>	1	1	0	0	0	0	1	1
<i>C</i>	0	1	0	0	0	0	0	1

<i>A</i>	1	0	1	0	1	0	1	0
<i>B</i>	1	1	0	0	0	0	1	1
<i>C</i>	1	0	0	0	0	0	1	0

vnútenie log. 0 do ľubovoľnej pozície v slove maskou (**B**). Výsledok logického súčinu bude mať log. **0** v bitoch, kde má log. **0** hodnotu operand **B**, v ostatných bitoch sa zachová pôvodná logická hodnota **X**

X X 0 0 0 0 X X

neekvivalencia

<i>A</i>	0	1	0	1	0	1	0	1
<i>B</i>	1	1	0	0	0	0	1	1
<i>C</i>	1	0	0	1	0	1	0	0

<i>A</i>	1	0	1	0	1	0	1	0
<i>B</i>	1	1	0	0	0	0	1	1
<i>C</i>	0	1	1	0	1	0	0	1

negácia log. hodnoty v ľubovoľnej pozícii v slove maskou (**B**). Výsledok XOR bude mať opačnú logickú hodnotu v bitoch, kde má log. **1** hodnotu operand **B**, v ostatných bitoch sa zachová pôvodná logická hodnota **X**

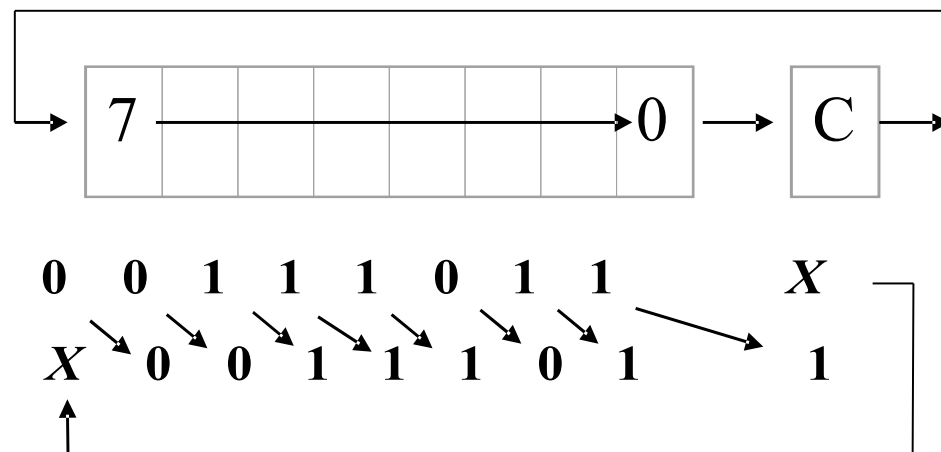
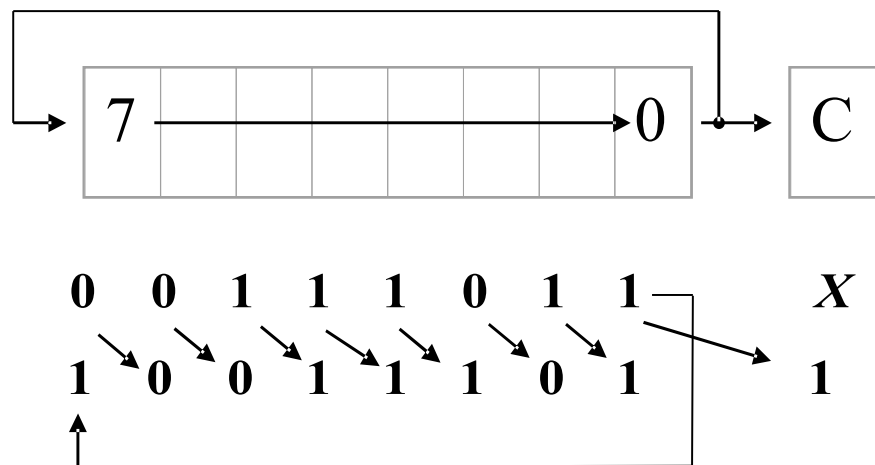
X XXX X X X X

Posuny a rotácie

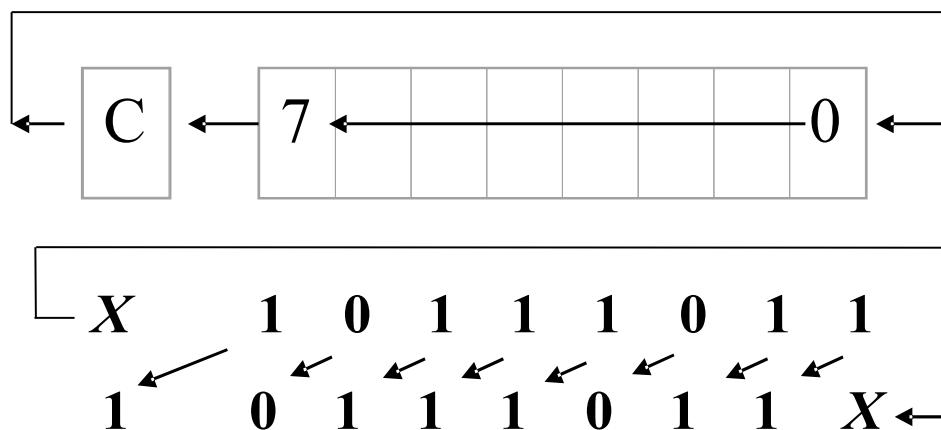
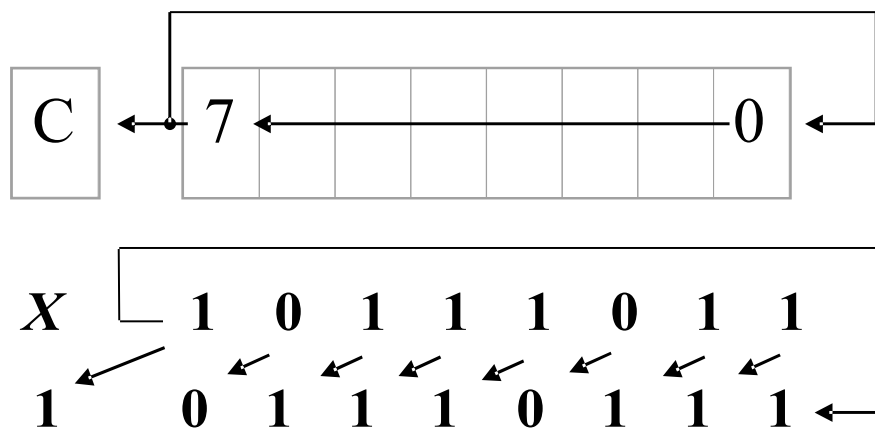
Posuny a **rotácie** obsahov pamäťových miest patria medzi **unárne** operátory.

Operácie využívajú pomocný bit – **príznakový bit CPU - Carry bit (C)**.

Rotácia obsahu pamäťového miesta **doprava**

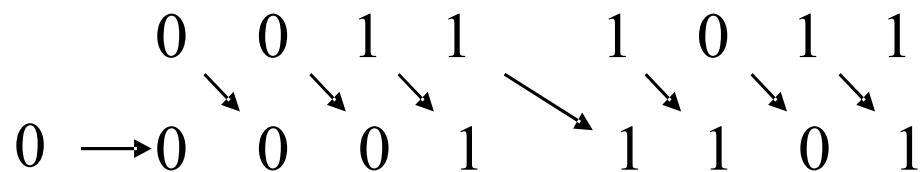
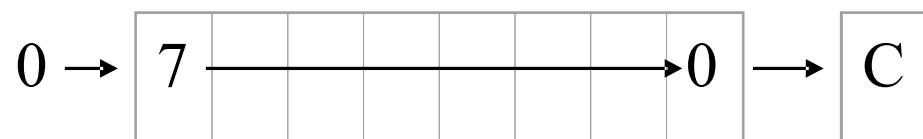


Rotácia obsahu pamäťového miesta **dol'ava**

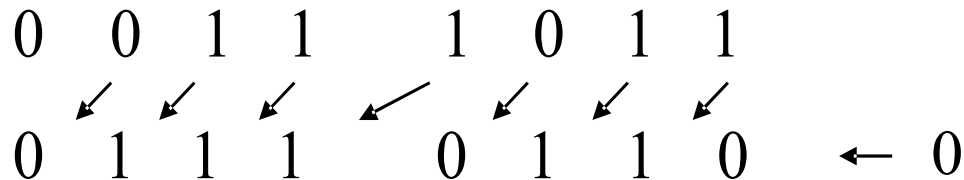
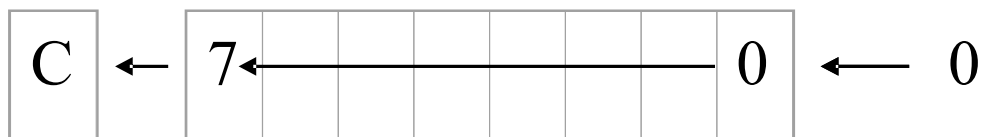


Logické posuny

Logický posun obsahu pamäťového miesta **doprava**

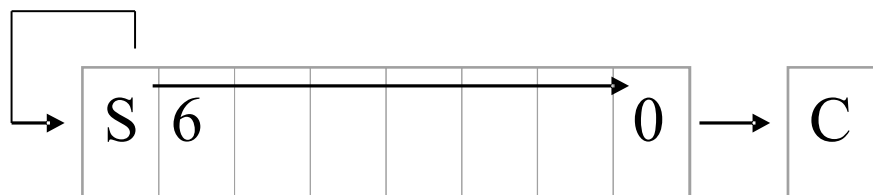


Logický posun obsahu pamäťového miesta **dol'ava**

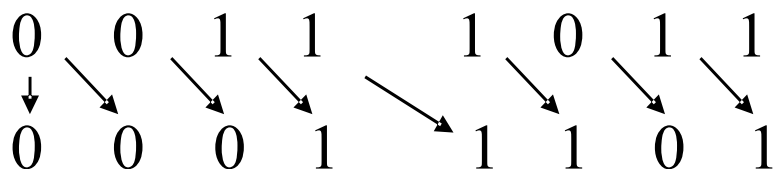


Aritmetické posuny

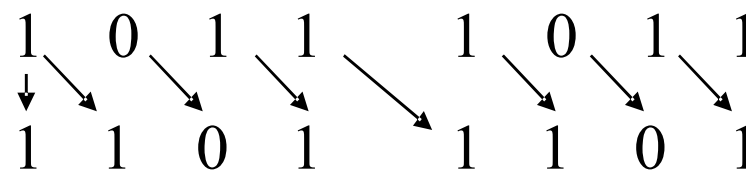
Aritmetický posun obsahu pamäťového miesta **doprava**



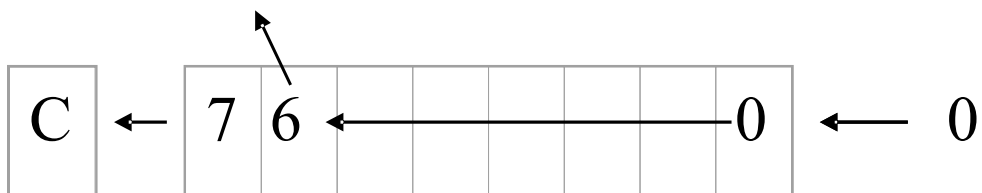
$S = 0$



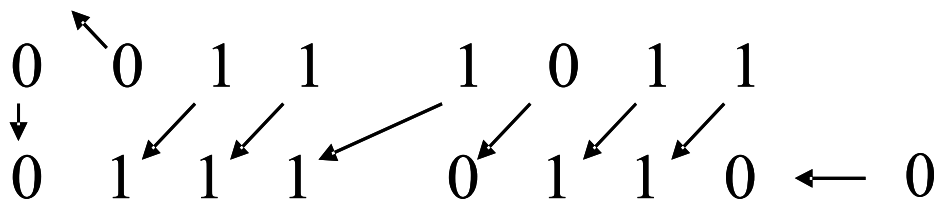
$S = 1$



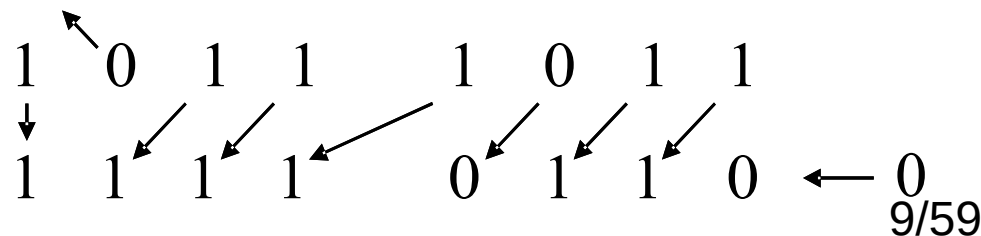
Aritmetický posun obsahu pamäťového miesta **dola**



$S = 0$

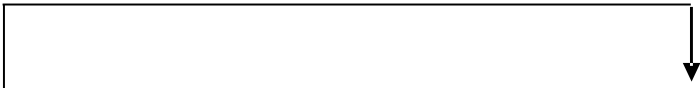


$S = 1$



Použitie:

- cyklické testovanie obsahu bitov
- vytváranie slov pri kódovaní
- aritmetické posuny
- aritmetický posun doprava je celočíselným delením číslom 2
- pri n posunoch ide o celočíselné delenie číslom 2^n


$$(0 \ 0 \ 1 \ 1 \ 1 \ 0 \ 1 \ 1)_2 = 32 + 16 + 8 + 2 + 1 = (59)_{10}$$

$$(0 \ 0 \ 0 \ 1 \ 1 \ 1 \ 0 \ 1)_2 = 16 + 8 + 4 + 1 = (29)_{10}$$

$$59 : 2 = 29 \quad R = 1 \quad a = b \gg n;$$

- celočíselný posun **doprava** je násobením číslom 2
- pri n posunoch ide o násobenie číslom 2^n

$$(0 \ 0 \ 1 \ 1 \ 1 \ 0 \ 1 \ 1)_2 = 32 + 16 + 8 + 2 + 1 = (59)_{10}$$

$$(0 \ 1 \ 1 \ 1 \ 0 \ 1 \ 1 \ 0)_2 = 64 + 32 + 16 + 4 + 2 = (118)_{10}$$

$$59 * 2 = 118 \quad a = b \ll n;$$

Jazyk C: Operátory na manipuláciu s bitmi

&	logický súčin po bitoch
	logický súčet po bitoch
^	neekvivalencia po bitoch
<<	posuv vľavo
>>	posuv vpravo
~	unárny operátor, jednotkový doplnok
!	negácia

Treba odlišovať **&**, **|** od logických spojok **&&** a **||**.

UNárny operátor -jeden parameter, pracuje s jedným údajom

BIárny operátor -dva parametre, pracuje s dvoma údajmi

TERnárny operátor -tri parametre, pracuje s tromi údajmi
?: v jazyku C.

Celočíselná aritmetika

Medzi základné celočíselné aritmetické operácie, ktoré realizuje ALU, patria

- súčet
- rozdiel
- násobenie
- delenie

Všetky tieto operácie sú binárne.

Binárne číslo:

- Dva znaky „0“ a „1“
- Znamienko mínus „-“
- „Desatinná“, binárna bodka „.“

Vieme zapísať: $(-1011.01010)_2 = (-11.3125)_{10}$

Počítač
pozná
len: „0“ a „1“

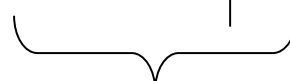
Celé čísla bez znamienka

Ak interpretujeme binárny obsah slova, ako zobrazenie dekadického čísla v pozičnej číselnej sústave so základom 2, ide o zobrazenie celých nezáporných čísel, t.j. celých čísel bez znamienka (sú to celé čísla typu **unsigned**).

Súčet, rozdiel čísel bez znamienka

Pre súčet, rozdiel binárnych čísel platia pravidlá binárnej aritmetiky pre jednobitové slová, kde okrem výsledného bitu, treba uvažovať signalizáciu prenosu, výpožičky do, z vyššieho rádu.

$S = A \begin{matrix} + \\ - \end{matrix} B$		+ -	
A	B	S	$carry (C)$ $borrow$
0	0	0	0 0
1	0	1	0 0
0	1	1	0 1
1	1	0	1 0

 Modulo 2, XOR

V prípade súčtu dvoch 8-bitových slov bude

$$\begin{array}{r}
 \begin{array}{cccccccc}
 & 0 & 0 & 1 & 1 & & 1 & 0 & 1 & 1 & (59)_{10} \\
 + & 0 & 1 & 1 & 1 & & 0 & 1 & 1 & 0 & (118)_{10} \\
 \hline
 \text{carry } \langle 0 \rangle & 1 & 1 & 1 & 1 & & 1 & 1 & 0 & & \\
 = & 1 & 0 & 1 & 1 & & 0 & 0 & 0 & 1 & (177)_{10}
 \end{array}
 \end{array}$$

59 + 118 = 177

v ďalšom príklade

$$\begin{array}{r}
 \begin{array}{cccccccc}
 1 & 0 & 1 & 1 & & 1 & 0 & 1 & 1 & (187)_{10} \\
 + & 0 & 1 & 1 & 1 & & 0 & 1 & 1 & 0 & (118)_{10} \\
 \hline
 \text{carry } \langle 1 \rangle & 1 & 1 & 1 & 1 & & 1 & 1 & 0 & & \\
 = & 0 & 0 & 1 & 1 & & 0 & 0 & 0 & 1 & (49)_{10}
 \end{array}
 \end{array}$$

$$187 + 118 = 305$$

V tomto prípade došlo k požiadavke na prenos do vyššieho rádu, po skončení sčítania. ALU obsahujú stavové slovo, ktorého jeden bit (Carry Bit) slúži na zapisovanie tejto informácie. Po vykonaní súčtu získame informáciu: **pamäťové miesto nestačí svojou veľkosťou na zápis výsledku sčítania.**

- pri viacslovnom sčítaní musíme uvažovať hodnotu tohto bitu pri sčítaní slov vyšších rádov
- v prípade súčtu celých čísel bez znamienka platí, že zároveň signalizuje tzv. pretečenie (**overflow**) (vo všeobecnosti neplatí).

Pretečenie znamená, že výsledok operácie nie je zobraziteľný (**to, čo je vo výsledku, nie je správny výsledok operácie**).

Na realizáciu sčítania sa v **ALU** používajú sčítačky realizované pomocou logických obvodov. Pri sčítaní na najnižšom ráde (sčítanie nultého rádu) sa používa **polovičná sčítačka**.

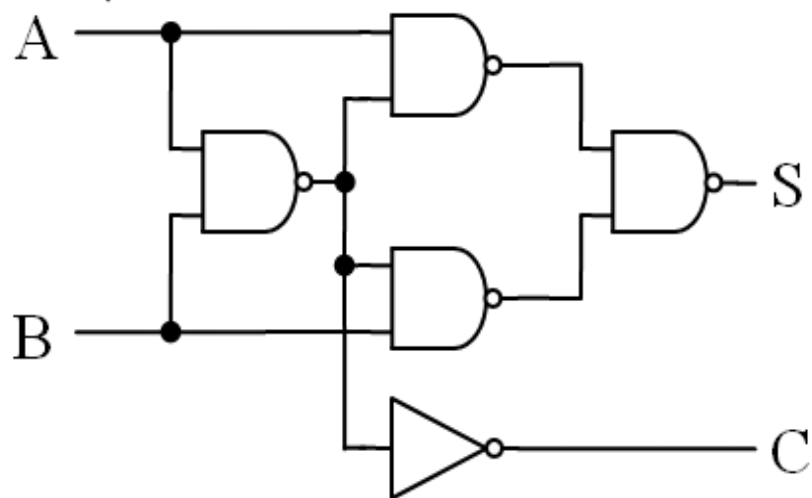
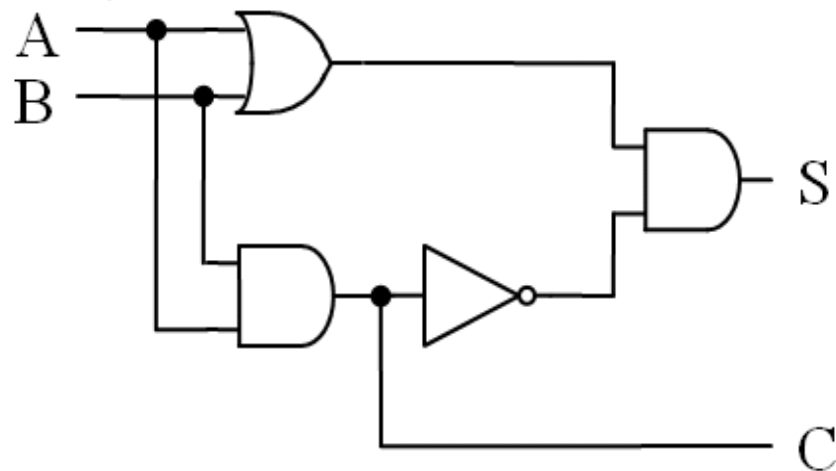
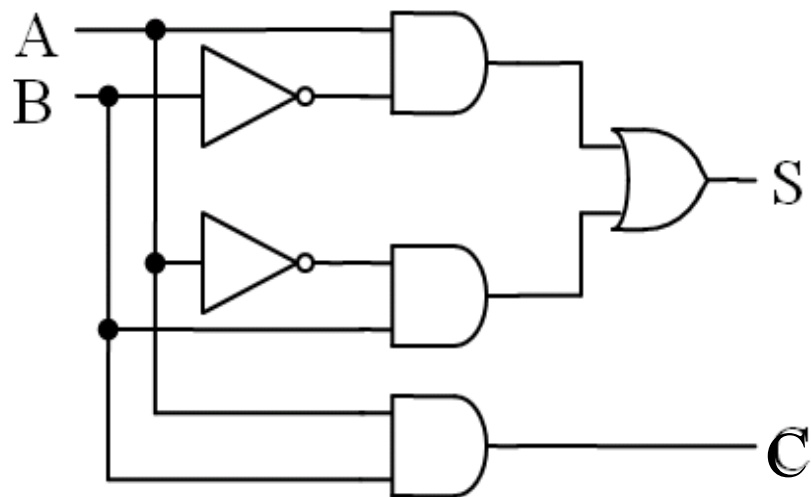
Popis **polovičnej sčítačky** pomocou aritmetických operácií

$$s_0 = (a_0 + b_0) \bmod 2 \quad c_1 = (a_0 + b_0) / 2$$

a logických operácií

$$s_0 = a_0 \bar{b}_0 + \bar{a}_0 b_0 \quad c_1 = a_0 \cdot b_0$$

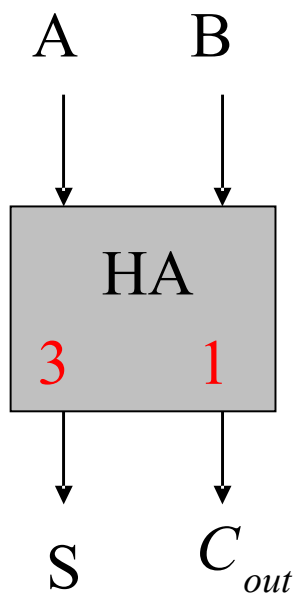
Polovičná sčítačka sa dá realizovať pomocou kombinačných obvodov



Tri možné spôsoby realizácie polovičnej sčítačky.

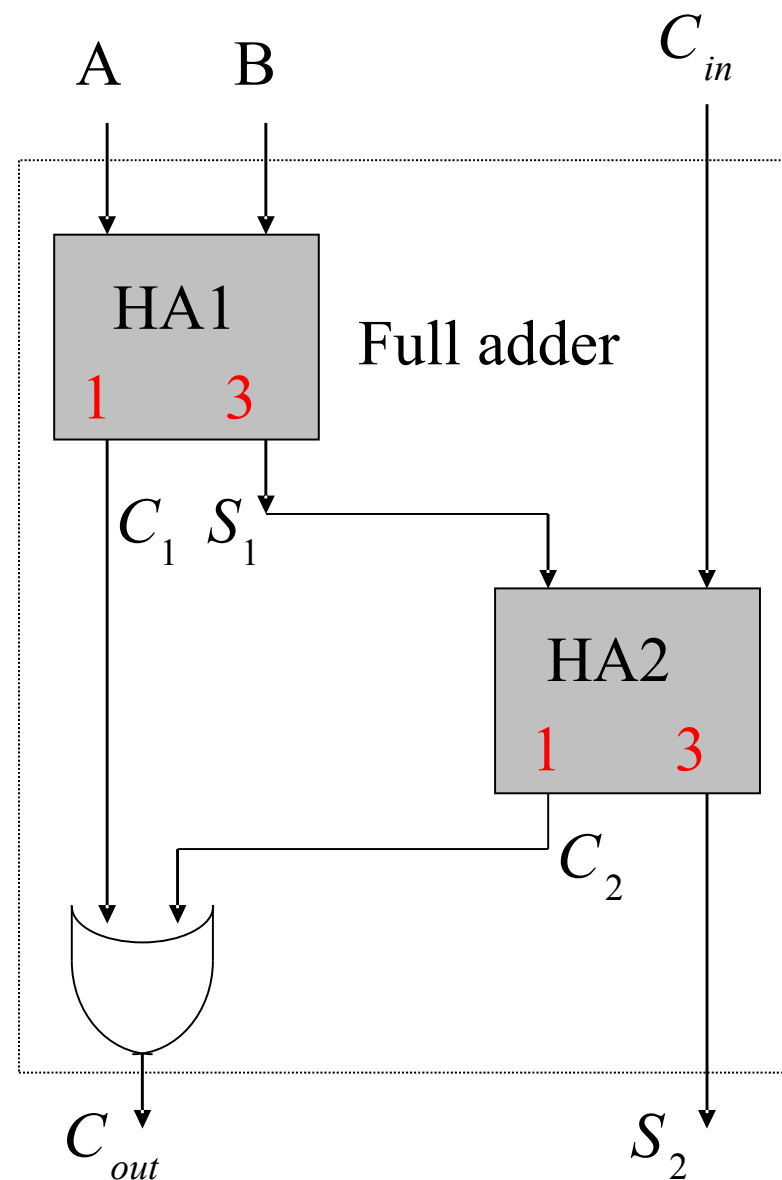
Oneskorenie: „S“ = 3 a „C“ = 1,2

Vytvorenie **úplnej sčítacky** pomocou polovičnej sčítacky :



Realizácia: 9 obvodov.

Oneskorenie: „S“ = 6 a „C“ = 5



Pri sčítaní na ďalších rádoch treba uvažovať prenosy z predchádzajúcich rádov

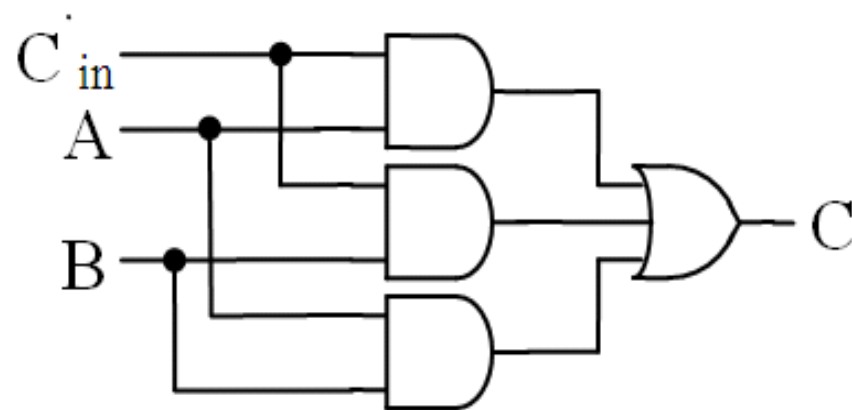
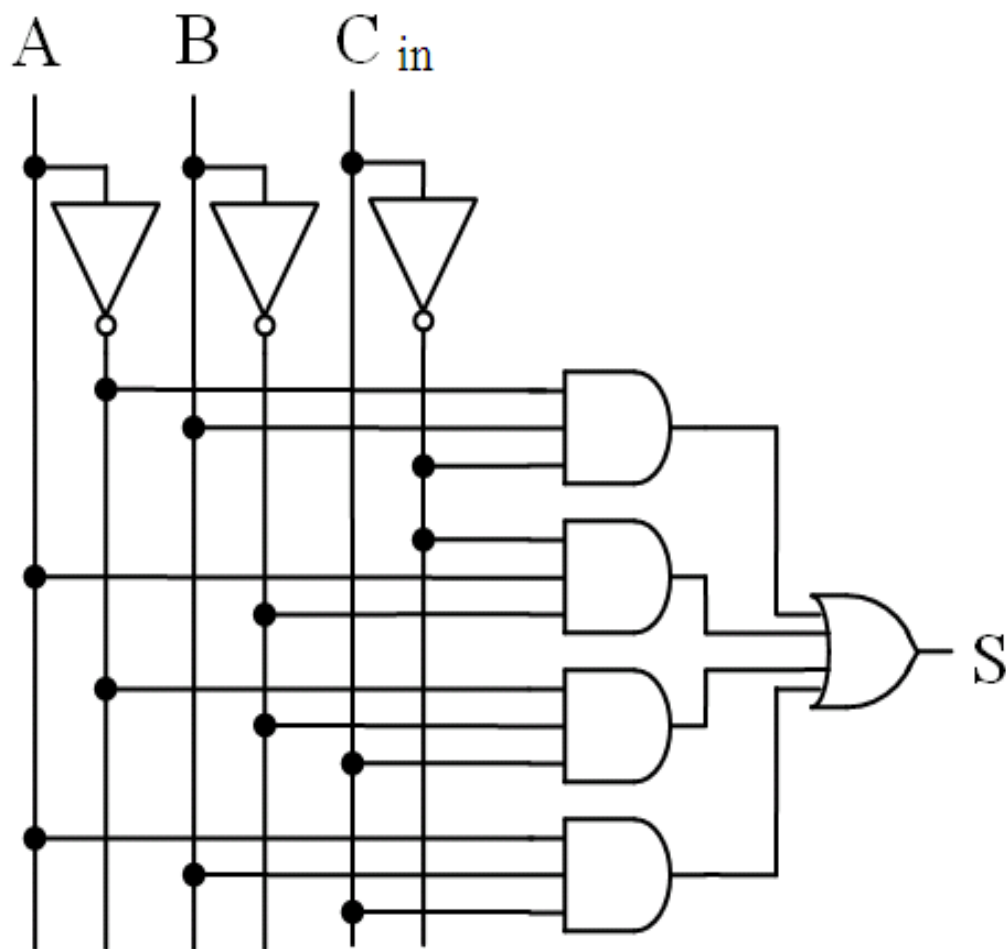
$$S = A + B$$

A	B	C_i	S	$carry (C)$
0	0	0	0	0
1	0	0	1	0
0	1	0	1	0
1	1	0	0	1
0	0	1	1	0
1	0	1	0	1
0	1	1	0	1
1	1	1	1	1

$$s_i = (a_i + b_i + c_i) \bmod 2 \quad c_{i+1} = (a_i + b_i + c_i) / 2$$

$$s_i = a_i \bar{b}_i \bar{c}_i + \bar{a}_i b_i \bar{c}_i + \bar{a}_i \bar{b}_i c_i + a_i b_i c_i \quad c_{i+1} = a_i b_i + (a_i + b_i) c_i$$

Možná praktická realizácia **úplnej sčítacky** :



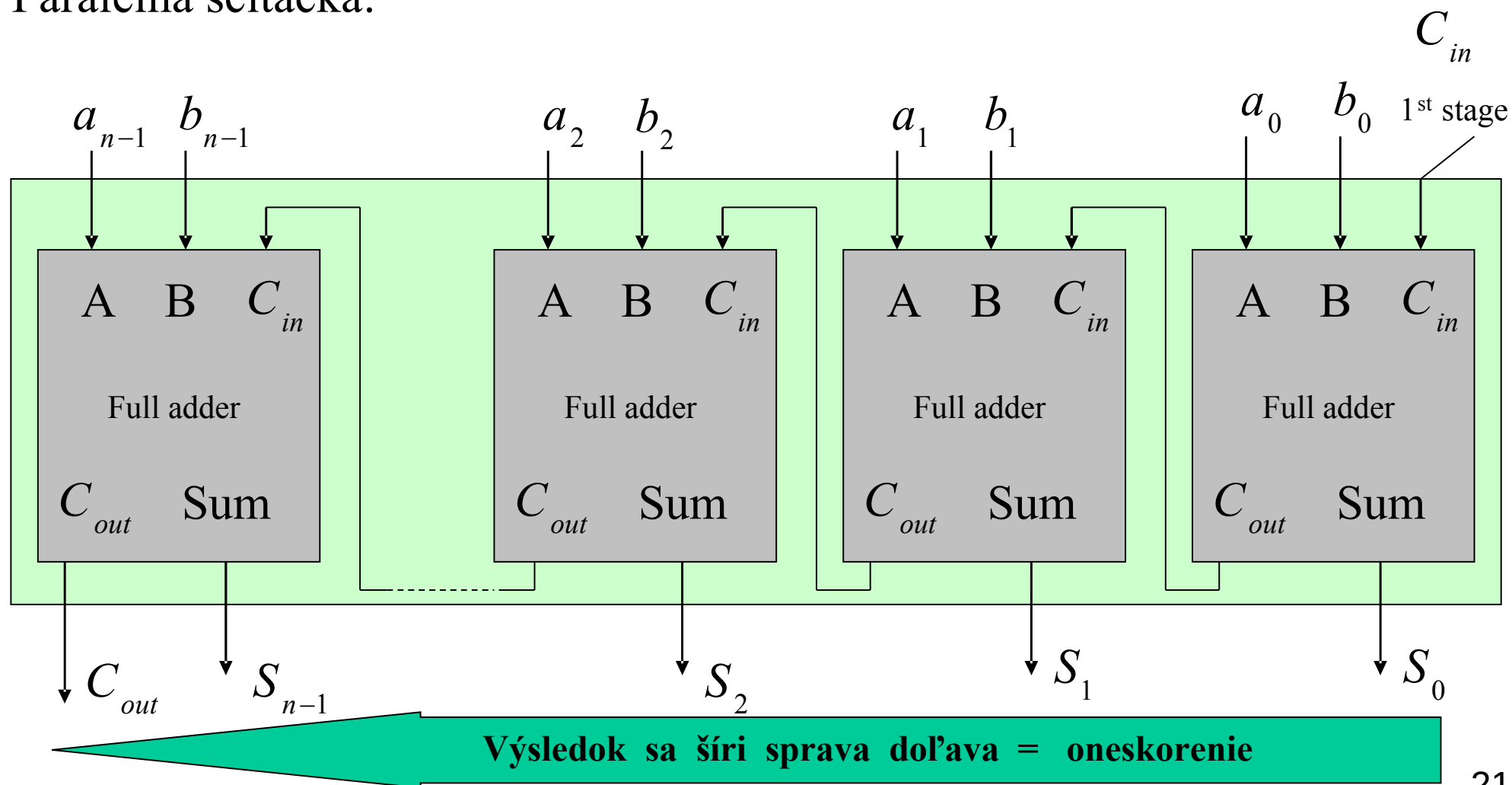
Realizácia: 12 obvodov

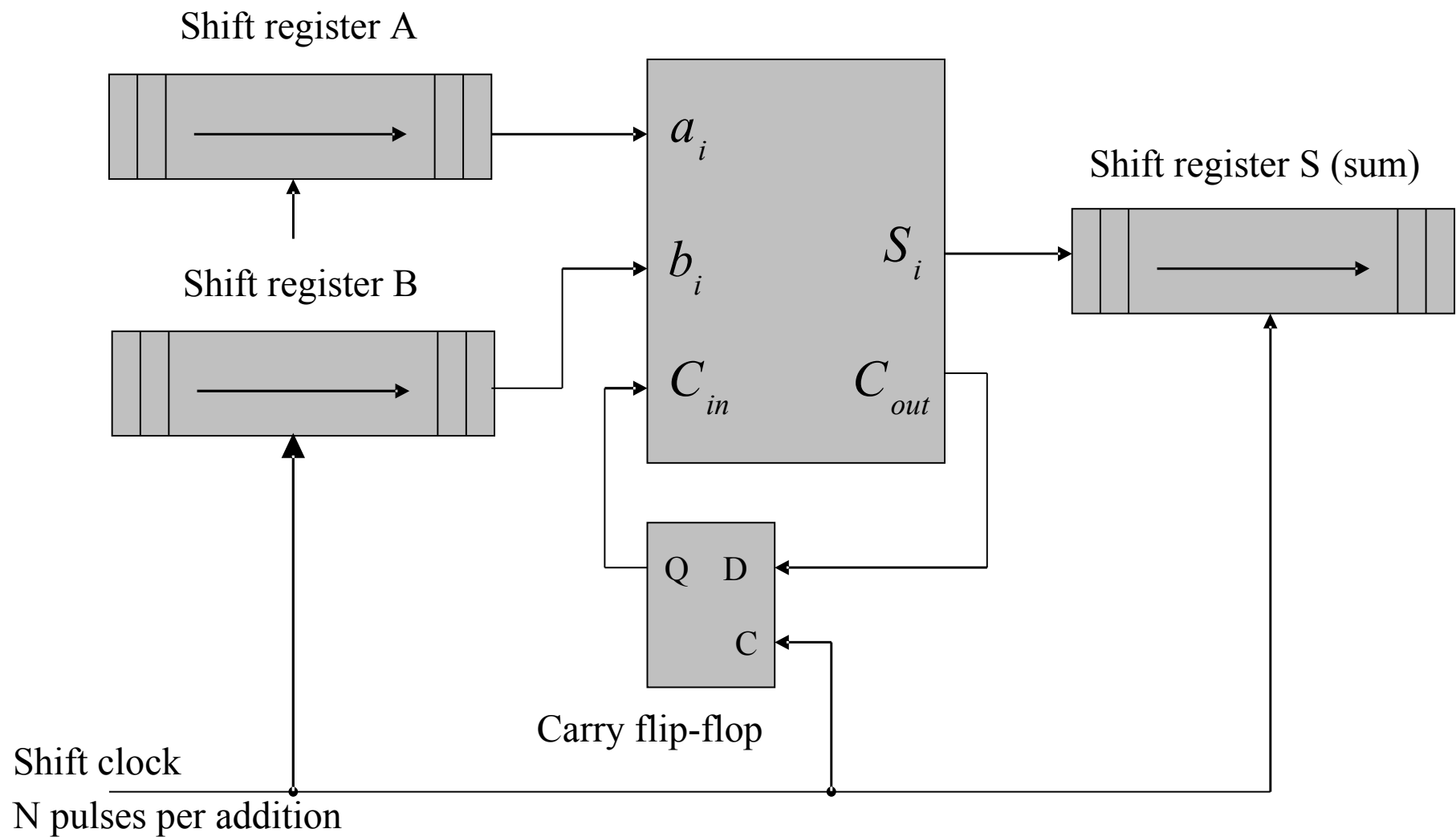
Oneskorenie: „S“ = 3 a „C“ = 2

Na sčítanie dvoch n-bitových čísel sa dajú použiť **paralelná** alebo **sériová** sčítačka, pričom sa vo všetkých stupňoch používa úplná sčítačka (realizácia prenosu z predchádzajúceho sčítania)

Inkrementácia: nastavíme carry bit a pričítame nuly

Paralelná sčítačka:





Sériová sčítačka: oneskorenie dané počtom taktov

Rozdiel čísel bez znamienka

Pre odčítanie binárnych čísel platia pravidlá binárnej aritmetiky pre jednobitové slová

$$S = A - B$$

A	B	S	$borrow (B)$
0	0	0	0
1	0	1	0
0	1	1	1
1	1	0	0

kde okrem výsledného bitu, treba uvažovať signalizáciu výpožičky z vyššieho rádu (borrow).

V prípade odčítania dvoch 8-bitových slov bude

$$\begin{array}{r}
 1 0 1 1 1 0 1 1 (187)_{10} \\
 - 0 1 1 1 0 1 1 0 (118)_{10} \\
 \hline
 \text{borrow } \langle 0 \rangle 1 0 0 0 1 0 0 0 \\
 0 1 0 0 0 1 0 1 (69)_{10}
 \end{array}$$

$$\begin{array}{r}
 0 0 1 1 1 0 1 1 (59)_{10} \\
 - 0 1 1 1 0 1 1 0 (118)_{10} \\
 \hline
 \text{borrow } \langle 1 \rangle 1 0 0 0 1 0 0 0 \\
 1 1 0 0 0 1 0 1 (197)_{10}
 \end{array}$$

$$59 - 118 = -122$$

$$[(256 + 59) - 118 = 197]$$

V druhom prípade je po vykonaní operácie nastavený príznak podtečenia

- v prípade viacbajtového odčítania je potrebné tento príznak uvažovať
- inak je výsledok nesprávny – výsledok nie je zobraziteľný

Hardvérovo sa dá odčítačka realizovať, ale na inom princípe, nie realizáciou pravidiel pre odčítanie.

Zobrazenie záporných celých čísel

Možné reprezentácie celých záporných čísel

- priamy kód
- inverzný
- doplnkový

Treba riešiť dve úlohy:

- Zobraziť číslo,
- Realizovať operáciu s týmto číslom

Priamy kód:

MSB bit, bit s najväčšou váhou: – znamienkový bit,

n-1 bitov: absolútna hodnota čísla

MSB = 0, kladné číslo $((-1)^0 = +1)$

MSB = 1, záporné číslo $((-1)^1 = -1)$

Napr.:

0 0 0 0 1 0 1 1 $(11)_{10}$

1 0 0 0 1 0 1 1 $(-11)_{10}$

- rozsah zobraziteľných čísel je symetrický $\langle -(2^{n-1} - 1), 2^{n-1} - 1 \rangle$
- pri aritmetických operáciách treba vyhodnocovať znamienka
- nejednoznačná nula

0 0 0 0 0 0 0 0 $(0)_{10}$

1 0 0 0 0 0 0 0 $(-0)_{10}$

Zložité aritmetické operácie (**potrebujeme sčítačku aj odčítačku**)

– nepoužíva sa

Inverzný kód (1's complement)

MSB bit, bit s najväčšou váhou: – znamienkový bit,

MSB = 0, kladné číslo

MSB = 1, záporné číslo

Číslo s opačným znamienkom sa vytvára inverziou bit po bite

$$0 \ 0 \ 0 \ 0 \quad 1 \ 0 \ 1 \ 1 \quad = (11)_{10}$$

$$1 \ 1 \ 1 \ 1 \quad 0 \ 1 \ 0 \ 0 \quad = (-11)_{10}$$

- rozsah zobraziteľných čísel je symetrický $\langle -(2^{n-1} - 1), 2^{n-1} - 1 \rangle$
- aritmetické operácie – zložité
- nejednoznačná nula

$$0 \ 0 \ 0 \ 0 \quad 0 \ 0 \ 0 \ 0 \quad (0)_{10}$$

$$1 \ 1 \ 1 \ 1 \quad 1 \ 1 \ 1 \ 1 \quad (-0)_{10}$$

- kód pre n-bitové záporné číslo je vytvorený podľa vzťahu

$$Y = -X \quad Y = 2^n - X - 1$$

Inverzný kód (1's complement)

n = 4

	x	x	x	x		x	x	x	x
0	0	0	0	0	-0	1	1	1	1
1	0	0	0	1	-1	1	1	1	0
2	0	0	1	0	-2	1	1	0	1
3	0	0	1	1	-3	1	1	0	0
4	0	1	0	0	-4	1	0	1	1
5	0	1	0	1	-5	1	0	1	0
6	0	1	1	0	-6	1	0	0	1
7	0	1	1	1	-7	1	0	0	0

$$Y = -X \quad Y = 2^n - X - 1$$

$$9 = 0000 \quad 1001 \quad -9 = 1111 \quad 0110$$

$$5 = 0000 \quad 0101 \quad -5 = 1111 \quad 1010$$

$$\begin{array}{r} -9 \quad 1111 \quad 0110 \\ +5 \quad 0000 \quad 0101 \\ \hline -4 \quad 1111 \quad 1011 \end{array} \quad \begin{array}{r} 9 \quad 0000 \quad 1001 \\ -5 \quad 1111 \quad 1010 \\ \hline 4 \quad 1 \quad 0000 \quad 0011 \end{array}$$

$$\begin{array}{r} -9 \quad 1111 \quad 0110 \\ +5 \quad 0000 \quad 0101 \\ \hline -4 \quad 1111 \quad 1011 \end{array} \quad \begin{array}{r} 9 \quad 0000 \quad 1001 \\ -5 \quad 1111 \quad 1010 \\ \hline 4 \quad 1 \quad 0000 \quad 0011 \end{array}$$

$$\begin{array}{r} -9 \quad 1111 \quad 0110 \\ +5 \quad 0000 \quad 0101 \\ \hline -4 \quad 1111 \quad 1011 \end{array} \quad \begin{array}{r} 9 \quad 0000 \quad 1001 \\ -5 \quad 1111 \quad 1010 \\ \hline 4 \quad 1 \quad 0000 \quad 0011 \end{array}$$

$$\begin{array}{r} -9 \quad 1111 \quad 0110 \\ +5 \quad 0000 \quad 0101 \\ \hline -4 \quad 1111 \quad 1011 \end{array} \quad \begin{array}{r} 9 \quad 0000 \quad 1001 \\ -5 \quad 1111 \quad 1010 \\ \hline 4 \quad 1 \quad 0000 \quad 0011 \end{array}$$

$$\begin{array}{r} -9 \quad 1111 \quad 0110 \\ +5 \quad 0000 \quad 0101 \\ \hline -4 \quad 1111 \quad 1011 \end{array} \quad \begin{array}{r} 9 \quad 0000 \quad 1001 \\ -5 \quad 1111 \quad 1010 \\ \hline 4 \quad 1 \quad 0000 \quad 0011 \end{array}$$

pri sčítaní treba **korigovať** výsledok pripočítaním obsahu carry bitu k LSB

Doplňkový kód (2's complement)

Bit s najväčšou váhou MSB – znamienkový bit, n-1 bitov
absolútna hodnota čísla

MSB = 0, kladné číslo

MSB = 1, záporné číslo

Číslo s opačným znamienkom sa vytvára v dvoch krokoch:

1) inverzia bit po bite

2) pripočítanie jednotky (inkrementácia)

$$\begin{array}{rcccccccccl} 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & (11)_{10} \\ 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & (negácia) \\ + & & & & & & & 1 & \\ \hline 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & (-11)_{10} \end{array}$$

$$\begin{array}{rcccccccccl}
 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & (-11)_{10} \\
 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & (negácia) \\
 + & & & & & & & 1 & \\
 \hline
 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & (11)_{10}
 \end{array}$$

z uvedeného vyplýva, že platí

$$X = -(-X)$$

- je len jedna nula

$$\begin{array}{rcccccccccl}
 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & (0)_{10} \\
 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & \\
 + & & & & & & & 1 & \\
 \hline
 \langle 1 \rangle & 0 & 0 & 0 & 0 & 0 & 0 & 0 & (-0)_{10}
 \end{array}$$

- výsledok sčítania dvoch čísel s ľubovoľnou kombináciou znamienok v doplnkovom kóde dáva správny výsledok

bez korekcie

9 = 0000 1001	− 9 = 1111 0111
5 = 0000 0101	− 5 = 1111 1011
4 = 0000 0100	− 4 = (1111 1011 + 1) = 1111 1100
− 9 1111 0111	9 0000 1001
+ 5 0000 0101	− 5 1111 1011
− 4 1111 1100	4 <1> 0000 0100

- v prípade súčtu čísel v doplnkovom kóde nastavenie carry bitu neznamená pretečenie.

V prípade nerovnakých znamienkových bitov (MSB) pretečeniu nemôže dôjsť.

Len v prípade, že MSB obidvoch operandov je rovnaké, pretečenie nastane vtedy, ak MSB výsledku sa nezhoduje s MSB operandov!

$$Overflow = MSB_A \cdot MSB_B \overline{MSB_S} + \overline{MSB_A} \cdot \overline{MSB_B} MSB_S$$

Problémom je, že hodnota MSB jedného z operandov v reálnom prípade je zmenená. Preto na detekciu overflow sa používajú prenosi do vyšších rádov (carry bity) posledných dvoch stupňov paralelnej sčítačky

$$Overflow = c_n \cdot \overline{c_{n-1}} + \overline{c_n} \cdot c_{n-1}$$

$$\begin{array}{rcl}
 & \begin{array}{c} \curvearrowright \\ \curvearrowright \end{array} & \\
 & \begin{array}{|c|c|} \hline 0 & 0 \\ \hline \end{array} & 00000110 \quad (+6) \\
 + & \begin{array}{|c|c|} \hline 0 & 0 \\ \hline \end{array} & 00001000 \quad (+8) \\
 \hline
 & & 00001110 \quad (+14)
 \end{array}$$

V=0, C=0 => OK

$$\begin{array}{rcl}
 & \begin{array}{c} \curvearrowright \\ \curvearrowright \end{array} & \\
 & \begin{array}{|c|c|} \hline 0 & 1 \\ \hline \end{array} & 11111111 \quad (+127) \\
 + & \begin{array}{|c|c|} \hline 0 & 0 \\ \hline \end{array} & 00000001 \quad (+1) \\
 \hline
 & & 10000000 \quad (-128)
 \end{array}$$


V=1, C=0 => ERROR

$$\begin{array}{rcl}
 & \begin{array}{c} \curvearrowright \\ \curvearrowright \end{array} & \\
 & \begin{array}{|c|c|} \hline 0 & 0 \\ \hline \end{array} & 00000010 \quad (+2) \\
 + & \begin{array}{|c|c|} \hline 1 & 1 \\ \hline \end{array} & 11111100 \quad (-4) \\
 \hline
 & & 11111110 \quad (-2)
 \end{array}$$

V=0, C=0 => OK

$$\begin{array}{rcl}
 & \begin{array}{c} \curvearrowright \\ \curvearrowright \end{array} & \\
 & \begin{array}{|c|c|} \hline 0 & 0 \\ \hline \end{array} & 00000100 \quad (+4) \\
 + & \begin{array}{|c|c|} \hline 1 & 1 \\ \hline \end{array} & 11111110 \quad (-2) \\
 \hline
 & & (C)00000010 \quad (+2)
 \end{array}$$

V=0, C=1 => OK

 = 0  = 1

$$\begin{aligned}
 V &= MSB_A \cdot MSB_B \overline{MSB_S} + \overline{MSB_A} \cdot \overline{MSB_B} MSB_S \\
 V &= c_n \cdot \overline{c_{n-1}} + \overline{c_n} \cdot c_{n-1}
 \end{aligned}$$

$$\begin{array}{rcl}
 & \begin{array}{c} \text{red} \\ \text{red} \end{array} & \\
 & \begin{array}{|c|c|} \hline 1 & 1 \\ \hline \end{array} & 1 \ 1 \ 1 \ 1 \ 0 \\
 + & \begin{array}{|c|c|} \hline 1 & 1 \\ \hline \end{array} & 1 \ 1 \ 1 \ 1 \ 0 \ 0 \\
 \hline
 \text{(C)} & 1 \ 1 \ 1 \ 1 & 1 \ 0 \ 1 \ 0 \\
 \end{array} \quad \begin{array}{l} (-2) \\ (-4) \\ (-6) \end{array}$$

V=0, C=~~1~~ => OK

$$\begin{array}{rcl}
 & \begin{array}{c} \text{red} \\ \text{green} \end{array} & \\
 & \begin{array}{|c|c|} \hline 1 & 0 \\ \hline \end{array} & 0 \ 0 \ 0 \ 0 \ 1 \\
 + & \begin{array}{|c|c|} \hline 1 & 1 \\ \hline \end{array} & 0 \ 0 \ 0 \ 1 \ 0 \\
 \hline
 \text{(C)} & 0 \ 1 & 0 \ 0 \ 0 \ 0 \ 1 \ 1 \\
 \end{array} \quad \begin{array}{l} (-127) \\ (-62) \\ (+67) \end{array}$$

V=1, C=1 => ERROR

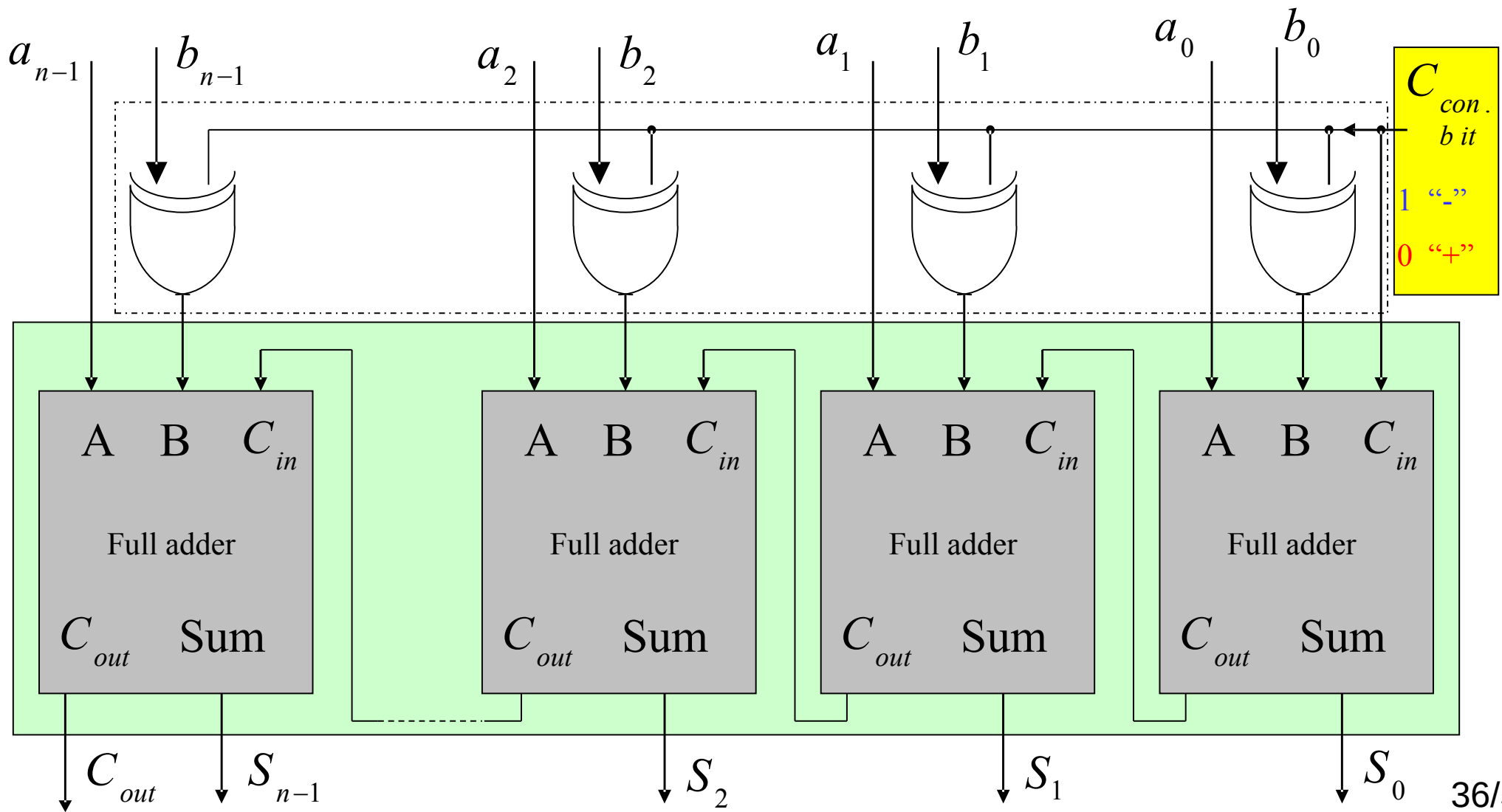
$$\begin{array}{c} \text{green} \end{array} = 0 \quad \begin{array}{c} \text{red} \end{array} = 1$$

$$V = c_n \cdot \overline{c_{n-1}} + \overline{c_n} \cdot c_{n-1}$$

$$V = MSB_A \cdot MSB_B \cdot \overline{MSB_S} + \overline{MSB_A} \cdot \overline{MSB_B} \cdot MSB_S$$

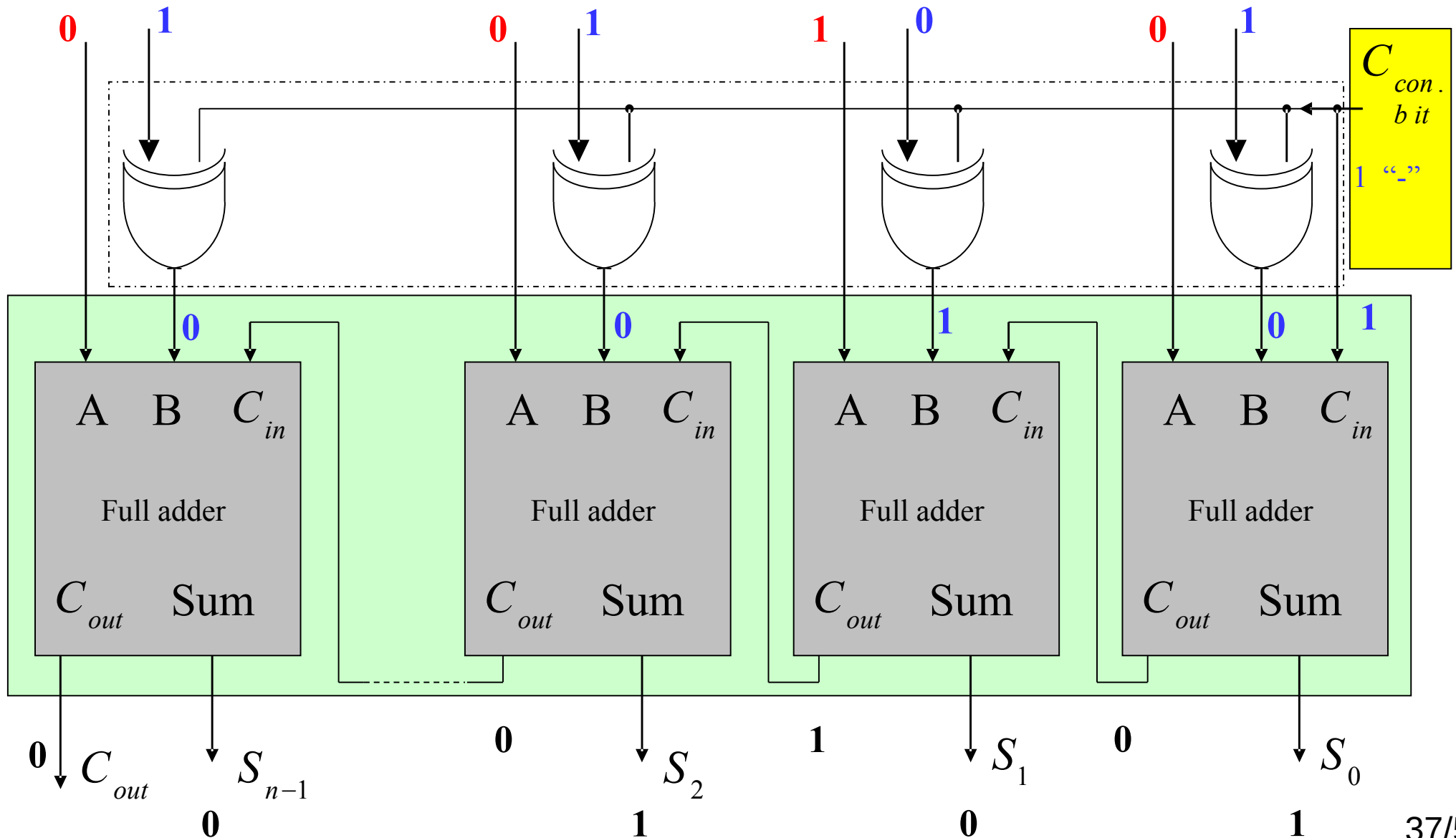
Sčítačka-odčítačka v doplnkovom kóde

- kód pre n-bitové záporné číslo v doplnkovom kóde je vytvorený podľa vzťahu: $Y = -X \quad Y = 2^n - X$
- rozsah zobraziteľných čísel je nesymetrický < -2



Príklad:

$$\{A = 2 (0010)\} - \{B = -3 (1101)\} = \{S = +5 (0101)\}$$



BCD kód

BCD (Binary Coded Decimal) – dvojkovo-desiatkové kódovanie

- zostáva desiatková sústava
- číslice sa kódujú do binárneho kódu

Existuje veľa možností kódovať dekadické číslice. Najväčší význam majú váhové kódy. Na kódovanie stačia 4 bity (nevyužívajú sa všetky kombinácie). Nech

$$d = a_3v_3 + a_2v_2 + a_1v_1 + a_0v_0$$

Aby kód bol vhodný pre aritmetické operácie mal by splniť nasledovné podmienky:

1. Jednoznačnosť – každej dekadickéj číslici je priradená jednoznačne jediná štvorica binárnych číslic
2. Aditívnosť – binárny súčet kódov číslic je rovný kódu ich súčtu
3. Symetričnosť – ak pre dve dekadické číslice platí: $d_i + d_j = 9$
potom medzi číslicami platí: $d_i = a_3 a_2 a_1 a_0 \Rightarrow d_j = \bar{a}_3 \bar{a}_2 \bar{a}_1 \bar{a}_0$
4. Usporiadanosť – väčším dekadickým čísliciam sú priradené väčšie kódy (možnosť porovnávania číslic)
5. Párnosť – párnym čísliciam zodpovedajú párne kódy, nepárnym nepárne

Najpoužívanější kód je pre :

$$v_3 = 8 \quad v_2 = 4 \quad v_1 = 2 \quad v_0 = 1$$

Binárny kód	Dekadická číslica
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010-1111	nevyužité kódy

Príklad:

Pre dekadické číslo $2357_{10} = 0010\ 0011\ 0101\ 0111$ BCD

Pozn.: Môžeme sa stretnúť s pojmami *zhustený* a *nezhustený* zápis čísel v BCD kóde. Posledne uvedený príklad zodpovedá zhustenému zápisu. V prípade nezhuseného kódu sa každá číslica kóduje do osobitného bajtu, doplnením núl do horných štyroch bitov

$$2357_{10} = \underline{\underline{0000\ 0010}} \quad \underline{\underline{0000\ 0011}} \quad \underline{\underline{0000\ 0101}} \quad \underline{\underline{0000\ 0111}}$$

Sčítanie a odčítanie v BCD kóde

Tento kód nie je symetrický, preto pri aritmetickom sčítaní a odčítaní je treba *upravovať* výsledok, v prípade **prenosu** do vyššieho rádu, alebo **výpožičke** z vyšších rádov.

Súčet dvoch operandov kódovaných v BCD kóde sa vykonáva rovnakým spôsobom ako súčet dvoch binárnych celých čísel, majme tri príklady :

Vykonajme súčet čísel kódovaných v BCD kóde

23+45,

23+29,

27+29

$$\begin{array}{r}
 23 \quad 0010 \quad 0011 \\
 + 45 \quad 0100 \quad 0101 \\
 \hline
 \quad \quad 0 \quad \quad < AC > \\
 68 \quad 0110 \quad 1000 \quad [68]_H
 \end{array}$$

V tomto prípade BCD kód výsledku sčítania zodpovedá BCD kódu správneho výsledku.

$$\begin{array}{r}
 23 \quad 0010 \quad 0011 \\
 + 29 \quad 0010 \quad 1001 \\
 \hline
 \quad \quad 0 \quad \quad < AC > \\
 52 \quad 0100 \quad 1100 \quad [4C]_H
 \end{array}
 \qquad
 \begin{array}{r}
 27 \quad 0010 \quad 0111 \\
 + 29 \quad 0010 \quad 1001 \\
 \hline
 \quad \quad 1 \quad \quad < AC > \\
 56 \quad 0101 \quad 0000 \quad [50]_H
 \end{array}$$

V prvom prípade je kód dolnej štvorice bitov nesprávny, v druhom prípade obidve číslice patria medzi správne BCD kódy, ale výsledok je nesprávny (prenos do vyššieho rádu medzi bitmi 3 a 4) .

$AC = Auxiliary\ Carry\ Flag$

Po vykonaní súčtu BCD čísel treba testovať

- **bud'** prenos do vyššieho rádu z každej štvorice bitov
- **alebo**, či kód číslice nepatrí do množiny kódov číslic 0 až 9
- alebo je výsledná číslica správna

Ak nastane jeden z dvoch prvých prípadov, vtedy vykonáme *opravu* pripočítaním čísla 6 na príslušnom ráde

$$\begin{array}{r}
 23 \quad 0010 \quad 0011 \\
 + 29 \quad 0010 \quad 1001 \\
 \hline
 \quad \quad 0 \quad \quad < AC >
 \end{array}$$

$$\begin{array}{r}
 52 \quad 0100 \quad 1100 \quad [4C]_H \\
 + 0000 \quad 0110 \quad [06]_H \\
 \hline
 0101 \quad 0010 \quad [52]_H
 \end{array}$$

$$\begin{array}{r}
 27 \quad 0010 \quad 0111 \\
 + 29 \quad 0010 \quad 1001 \\
 \hline
 \quad \quad 1 \quad \quad < AC >
 \end{array}$$

$$\begin{array}{r}
 56 \quad 0101 \quad 0000 \quad [50]_H \\
 + 0000 \quad 0110 \quad [06]_H \\
 \hline
 0101 \quad 0110 \quad [56]_H
 \end{array}$$

V obidvoch prípadoch sme získali po korekcii správny kód výsledku.

V prípade odčítania môže nastať rovnaká situácia ako pri sčítaní, že dôjde k podtečeniu medzi štvoricami bitov alebo k nesprávnemu kódu číslice.

$$\begin{array}{r}
 53 \quad 0101 \quad 0011 \\
 - 29 \quad 0010 \quad 1001 \\
 \hline
 1 < AC > \\
 24 \quad 0010 \quad 1010 \quad 2A \\
 - 0000 \quad 0110 \quad 06 \\
 \hline
 0010 \quad 0100 \quad 24
 \end{array}$$

V tomto prípade vo výsledku je neplatný kód číslice s najnižšou váhou a došlo k podtečeniu medzi bitmi 3 a 4.

Súčin celých čísel bez znamienka

Majme dva binárne osembitové operandy

$$A = a_7 2^7 + a_6 2^6 + \dots + a_1 2^1 + a_0 2^0$$

$$B = b_7 2^7 + b_6 2^6 + \dots + b_1 2^1 + b_0 2^0$$

$$S = A.B = A.(b_7 2^7 + b_6 2^6 + \dots + b_1 2^1 + b_0 2^0) =$$

$$b_7 A 2^7 + b_6 A 2^6 + \dots + b_1 A 2^1 + b_0 A 2^0$$

Príklad :

$$\begin{array}{r}
 \begin{array}{cccc} 1 & 0 & 1 & 1 \end{array} (11)_{10} \\
 * \begin{array}{cccc} 0 & 1 & 1 & 0 \end{array} (6)_{10} \\
 \hline
 \begin{array}{ccccccccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 * (11)_{10} * 2^0 \\
 + & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 (11)_{10} * 2^1 \\
 \hline
 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\
 + & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 (11)_{10} * 2^2 \\
 \hline
 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 64 + 2 = 66
 \end{array}
 \end{array}$$

Ak sú operandy n-bitové, výsledok bude 2n-bitový.

Súčin celých čísel (so znamienkom)

Budeme uvažovať len čísla vyjadrené v doplnkovom kóde. Na to aby algoritmus násobenia bol použiteľný, je treba:

- previesť operandy na kladné čísla, a potom
- upraviť výsledok podľa výsledného znamienka súčinu a
- korigovať výsledok získaný popísaným algoritmom.

Majme operandy X a $-Y$, v doplnkovom kóde $-Y = 2^n - Y$ potom

$$X(-Y) \rightarrow X(2^n - Y) = 2^n X - XY$$

obrazom $-XY$ je v doplnkovom kóde

$$X(-Y) \rightarrow 2^{2n} - XY$$

potom musíme korigovať výsledok pripočítaním čísla

$$\begin{aligned} \textit{korekcia} &= (2^{2n} - XY) - (2^n X - XY) = 2^{2n} - 2^n X = \\ &= 2^n (2^n - X) \end{aligned}$$

čo je vlastne o n-bitov posunutý operand $(-X)$ zobrazený v doplnkovom kóde, ktorý treba pripočítať k súčinu $X(-Y)$.

Príklad : Majme čísla 15 a -13 . Na ich zobrazenie potrebuje 5 bitov 1 bit znamienkový a 4 bity na zobrazenie čísel 13 a 15. Výsledok násobenia týchto čísel bude 10 bitový.

$$15 = 01111_2 \quad 13 = 01101_2$$

$$-15 = 10000 + 1 = 10001$$

$$-13 = 10010 + 1 = 10011$$

9 8 7 6 5 4 3 2 1 0

0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 1 1 1 1

0 0 0 0 0 1 1 1 1 0

0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0

0 0 1 1 1 1 0 0 0 0

0 1 0 0 0 1 1 1 0 1 (285)

1 1 0 0 1 1 1 1 0 1 $\leftarrow (-195)$

0 0 1 1 0 0 0 0 1 0 + 1


0 0 1 1 0 0 0 0 1 1 = 128 + 64 + 3 = 195

$$X(-Y) = 15 * (-13)$$

Podobná situácia nastane v prípade, že sú obidva operandy záporné

$$(-X)(-Y) \rightarrow (2^n - X)(2^n - Y) = 2^{2n} - 2^n X - 2^n Y + XY$$

Aby bol výsledok správny musíme pripočítať k výsledku $2^n \cdot X$ a $2^n \cdot Y$, a ignorovať jednotku v Carry Bite.


$$korekcia = (XY) - (2^{2n} - 2^n X - 2^n Y + XY) = -2^{2n} + 2^n X + 2^n Y$$

Booth-ov algoritmus

Algoritmus dáva správny výsledok pre všetky kombinácie kladných a záporných operandov.

V tomto prípade sa testujú dva bity násobiteľ'a naraz, okrem aktuálneho bitu sa testuje najbližší nižší bit (môžu nastať tri situácie) :

1. v aktuálnom bite je 1 a nasledujúcom je 0 – potom odčítame násobenca od výsledku
2. v aktuálnom bite je 0 a nasledujúcom je 1 – potom pripočítame násobenca k výsledku
3. ak sú uvedené bity rovnaké nerobíme nič
4. ak pri pripočítaní sa nastaví carry bit, ten sa ignoruje
5. ak je testované LSB násobiteľ'a, nižší bit sa predpokladá, že je nulový
6. pri posune medzivýsledku sa používa aritmetický posun (kopíruje sa MSB)

$$15 = 01111_2 \quad -15 = 10000 + 1 = 10001$$

$$13 = 01101_2 \quad -13 = 10010 + 1 = 10011$$

$$15 * 13 = 195$$

C	9	8	7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	0	0	0	0	<u>0 1 1 0 1</u> 0* (-)
0	<u>1</u>	0	0	0	1	0	0	0	0	0	
0	1	0	0	0	1	0	0	0	0	0	(→)
0	1	1	0	0	0	1	0	0	0	0	0 1 1 <u>0</u> 1 (+)
0	<u>0</u>	1	1	1	1	0	0	0	0	0	
1	0	0	1	1	1	1	0	0	0	0	(→)
0	0	0	0	1	1	1	1	0	0	0	0 1 <u>1</u> 0 1 (-)
0	<u>1</u>	0	0	0	1	0	0	0	0	0	
0	1	0	1	0	0	1	1	0	0	0	(→)
0	1	1	0	1	0	0	1	1	0	0	0 <u>1</u> 1 0 1 (→)
0	1	1	1	0	1	0	0	1	1	0	<u>0</u> 1 1 0 1 (+)
0	<u>0</u>	1	1	1	1	0	0	0	0	0	
1	0	1	1	0	0	0	0	1	1	0	(→)
1	0	0	1	1	0	0	0	0	1	1	= 128 + 64 + 3 = 195

$$15 = 01111_2 \quad -15 = 10000 + 1 = 10001$$

$$13 = 01101_2 \quad -13 = 10010 + 1 = 10011$$

$$15 * (-13) = -195$$

C 9 8 7 6 5 4 3 2 1 0

0 0 0 0 0 0 0 0 0 0 0

1 0 0 1 1 0* (-)

0 1 0 0 0 1 0 0 0 0 0

0 1 0 0 0 1 0 0 0 0 0

(→)

0 1 1 0 0 0 1 0 0 0 0

1 0 0 1 1 (→)

0 1 1 1 0 0 0 1 0 0 0

1 0 0 1 1 (+)

0 0 1 1 1 1 0 0 0 0 0

1 0 1 0 1 1 0 1 0 0 0

(→)

0 0 0 1 0 1 1 0 1 0 0

1 0 0 1 1 (→)

0 0 0 0 1 0 1 1 0 1 0

1 0 0 1 1 (-)

0 1 0 0 0 1 0 0 0 0 0

0 1 0 0 1 1 1 1 0 1 0

(→)

0 1 1 0 0 1 1 1 1 0 1

$$= -(0011000010 + 1) = -(0011000011) = -195$$

Idea Booth-ovho algoritmu

$$A * (00111110) = A * (2^5 + 2^4 + 2^3 + 2^2 + 2^1) = A * (2^6 - 2^1)$$

$$2^5 + 2^4 + 2^3 + 2^2 + 2^1 = 32 + 16 + 8 + 4 + 2 = 62$$

$$2^6 - 2^1 = 64 - 2 = 62$$

Pri násobení namiesto piatich súčtov operandu A s medzivýsledkom násobenia, stačí jeden súčet a jeden rozdiel posunutého operandu A.

Dá sa ukázať podobná vlastnosť v prípade násobenia čísel kódovaných v doplnkovom kóde.

Podiel celých čísel bez znamienka

Pre celočíselné delenie platí

$$\frac{X}{Y} = P + \frac{Z}{Y} \quad \text{alebo} \quad X = P.Y + Z$$

$$575 : 25 = 23$$

$$\begin{array}{r} - \underline{50} \\ 75 \\ - \underline{75} \\ 0 \end{array}$$

$$575 : 45 = 12$$

$$\begin{array}{r} - \underline{45} \\ 125 \\ - \underline{90} \\ 35 \end{array}$$

$$575 = 1000111111_2 \quad 25 = 011001_2 \quad -25 = 100111_2$$

$$575 : 25 = 23$$

$$\begin{array}{cccccccccc} 9 & 8 & 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 \end{array}$$

$$\begin{array}{cccccccccc} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \end{array}$$

$$\begin{array}{cccccccccc} 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \end{array} (+)$$

$$\begin{array}{cccccccccc} 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \end{array} (D > 0 \quad P = 1)$$

$$\begin{array}{cccccccccc} 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \end{array} (\rightarrow)$$

$$\begin{array}{cccccccccc} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 \end{array} (D < 0 \quad P = 10)$$

$$\begin{array}{cccccccccc} 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \end{array} ()$$

$$\begin{array}{cccccccccc} 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \end{array} (+)$$

$$\begin{array}{cccccccccc} 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{array} (D > 0 \quad P = 101)$$

$$\begin{array}{cccccccccc} 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \end{array} (+)$$

$$\begin{array}{cccccccccc} 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \end{array} (D > 0 \quad P = 1011)$$

$$\begin{array}{cccccccccc} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 \end{array}$$

$$\begin{array}{cccccccccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array} (D > 0 \quad P = 10111) \quad (\text{zvyšok} = 0)$$

$$10111_2 = 16 + 4 + 2 + 1 = 23$$

Príznakové bity

Sú súčasťou príznakového registra (stavového slova procesora), ktorý agreguje okrem iného aj jednobitové informácie o výsledku aritmeticko-logickej operácie, medzi ktoré patria :

- (C), (CY) (**Carry**) (signalizácia prenosu do vyššieho rádu/ výpožičky z vyššieho rádu)
- (Z) (**Zero**) príznak nuly (všetky bity výsledku sú nulové)
- (S) (**Signum**) kopíruje MSB výsledku
- **Overflow**, Parity atď.

Význam príznakových bitov pri vetvení programu !!

Literatúra:

- [1] Clements,A: The Principles of Computer Hardware, Oxford
- [2] Stalling, W.: Computer Organization and Architecture,
principles ...,