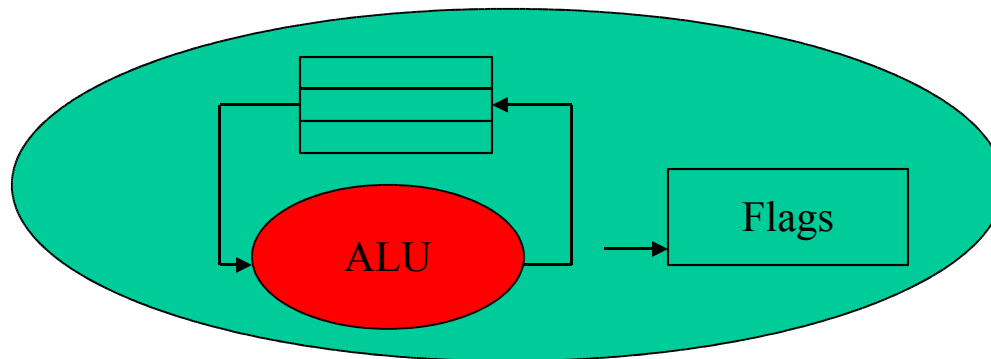


Základné aritmeticko-logické operácie

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**)



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

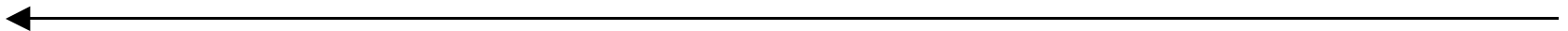
Označovanie bitov v 8-bitovom slove (v jednobajtovom slove)

7 (MSB)	6	5	4	3	2	1	0 (LSB)
---------	---	---	---	---	---	---	---------



Označovanie bitov v 16-bitovom slove (v dvojbajtovom slove)

15 (MSB)	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0 (LSB)
----------	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---------



LSB – Least Significant Bit (bit s najmenšou váhou)

MSB – Most Significant Bit (bit s najväčšou váhou)

Bit (1b), Nibble (4b), Byte (8b=1B), Word (2B,)

Operácie delíme ich na

- **unárne** – vykonávajú sa s jedným pamäťovým miestom
- **binárne** – vykonávajú sa s dvomi pamäťovými miestami

Medzi elementárne operácie, ktoré vykonávajú takmer všetky CPU, patria :

- negácia
- logický súčet
- logický súčin
- neekvivalencia

Negácia (NOT)

Patrí medzi unárne operácie

Pravidlá pre jednobitové operandy sú

$$B = NOT(A) \quad [B = \overline{A}]$$

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

Pre viacbitové (budeme uvádzať príklady pre 8-bitové operandy (jeden bajt)

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

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

Logický súčet (OR)

Patrí medzi binárne operácie

Pravidlá pre jednobitové operandy sú

$$C = A \text{ OR } B \quad [C = A + B]$$

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

Pre jednobajtové operandy

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

<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

Použitie – vnútenie logickej jednotky do ľubovolnej pozície v byte (bytoch) tzv. maskou

Napr. Vykonaním logického súčtu ľubovольného bytu s maskou, ktorá je reprezentovaná bytom s obsahom

1 1 0 0 0 1 0 0 [$C4_H$]

bude v uvedenom byte v bitoch s indexom 7,6 a 2 vždy logická jednotka bez ohľadu na predchádzajúci obsah bytu.

1 1 X X X 1 X X

kde X je obsah bitu pred vykonaním operácie.

Logický súčin (AND)

Patrí medzi binárne operácie

Pravidlá pre jednobitové operandy sú

$$C = A \text{ AND } B \quad [C = A.B]$$

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

Pre jednobajtové operandy

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

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

Použitie

– vnútenie logickej nuly do ľubovolnej pozície v byte
(bytech) tzv. maskou

Napr.: Vykonaním logického súčinu ľubovolného bytu
s maskou, ktorá je reprezentovaná bytom s obsahom

$$0 \ 0 \ 1 \ 1 \ 1 \ 0 \ 1 \ 1 \ [3B_H]$$

bude v uvedenom byte v bitoch s indexom 7,6 a 2 vždy logická nula bez ohľadu na predchádzajúci obsah bitu.

0 0 X X X 0 X X

- pomocou masky môžeme testovať logickú hodnotu vybraného bitu (bitov) bajtu (bajtov).

Použitím napr. masky

0 0 0 0 0 0 1 0 [02_H]

po vykonaní logického súčinu s vybraným bajtom, bude výsledok operácie nulový, ak v bite s indexom 1 bola logická nula, a naopak nenulový, ako v danom bite bola logická jednotka.

Nulový výsledok znamená, že bajt (bajty), do ktorých sa zapíše výsledok aritmeticko-logickej operácie obsahuje vo všetkých bitoch binárnu hodnotu nula.

Neekvivalencia (XOR)

Patrí medzi binárne operácie

Nazýva sa tiež exkluzívny súčet, súčet modulo 2, EOR

Pravidlá pre jednobitové operandy sú

$$C = A \text{ XOR } B \quad [C = A \oplus B]$$

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

Pre jednobajtové operandy

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

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

Použitie

– invertovanie logickej hodnoty v ľubovolnej pozícii v bajte (bajtoch) maskou

Napr. Vykonaním neekvivalencie ľubovolného bytu s maskou, ktorá je reprezentovaná bytom s obsahom

$$0 \ 0 \ 1 \ 1 \quad 1 \ 0 \ 1 \ 1 \quad [3B_H]$$

bude výsledok operácie

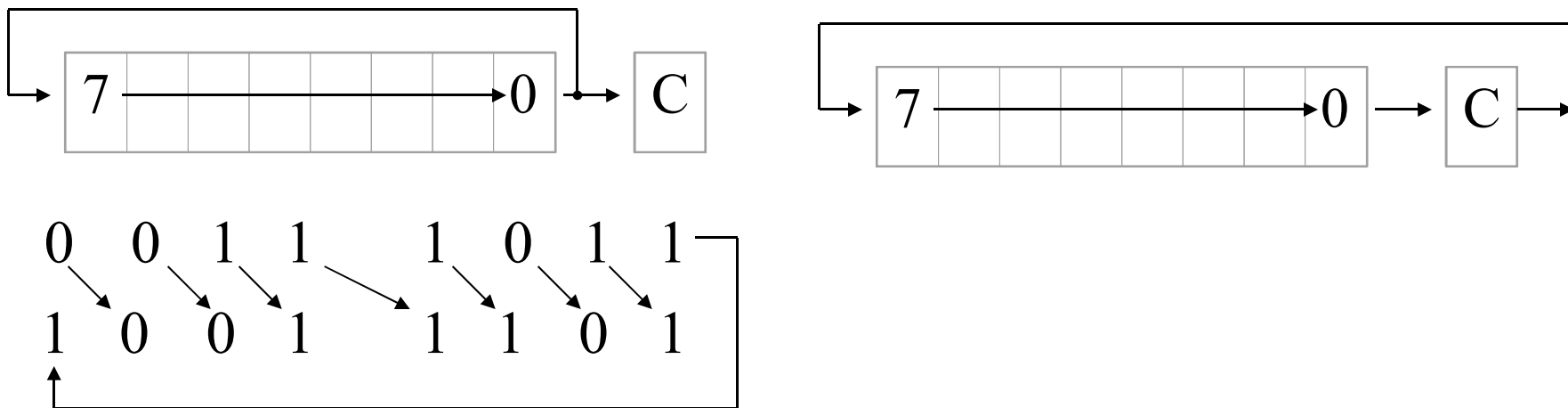
$$X \ X \ \bar{X} \ \bar{X} \quad \bar{X} \ X \ \bar{X} \ \bar{X}$$

Posuny a rotácie

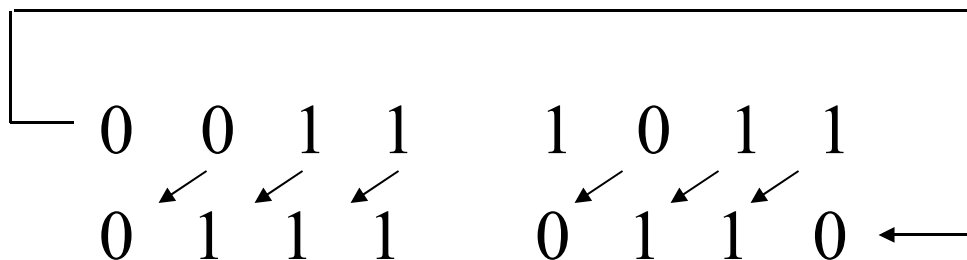
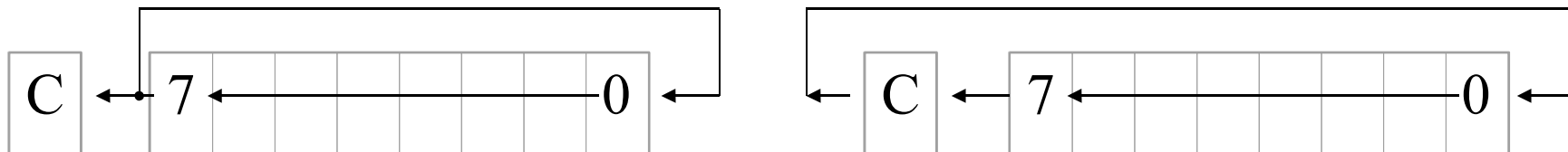
Medzi najzákladnejšie operácie ALU patria posuny a rotácie obsahov pamäťových miest. Patria medzi unárne operátory. Operácie sa vykonávajú posúvaním obsahov bitov doľava, alebo doprava, prípadne n-tica pamäťových bitov sa doplní formálne o pomocný bit na $(n+1)$ -tícu (pomocným bitom bývajú príznakové (stavové) bity CPU, napr. Carry bit)

Rotácie

Rotácia obsahu pamäťového miesta **doprava**



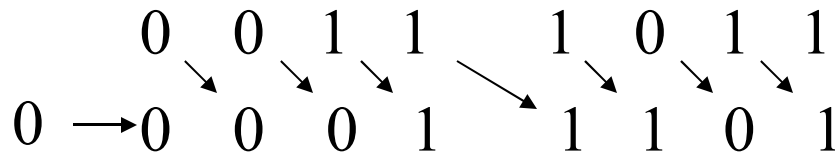
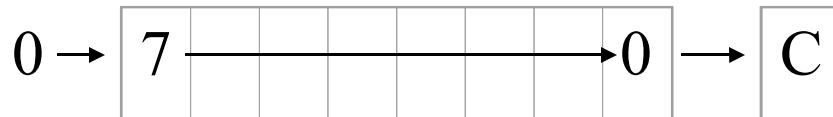
Rotácia obsahu pamäťového miesta **dol'ava**



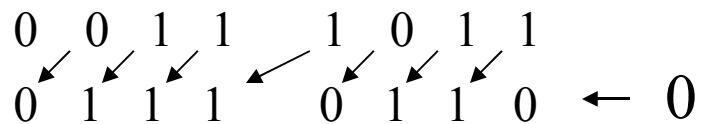
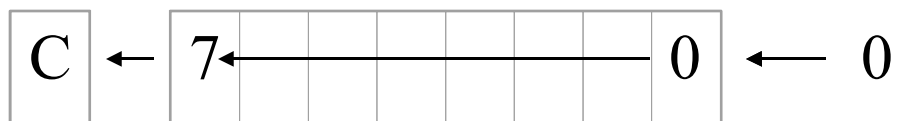
Posuny obsahu pamäťového miesta môžu byť
- **logické** alebo **aritmetické**

Logické posuny

Logický posun obsahu pamäťového miesta **doprava**

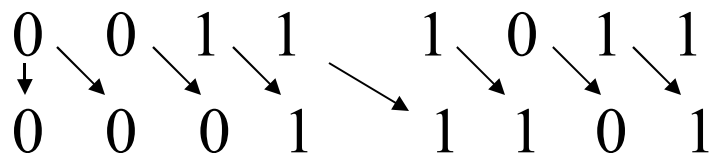
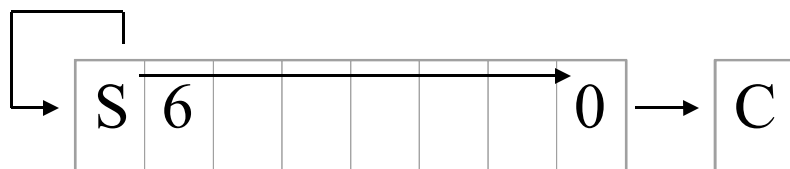


Logický posun obsahu pamäťového miesta **dol'ava**

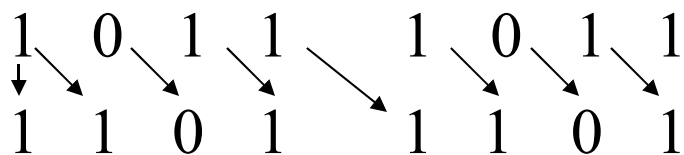


Aritmetické posuny

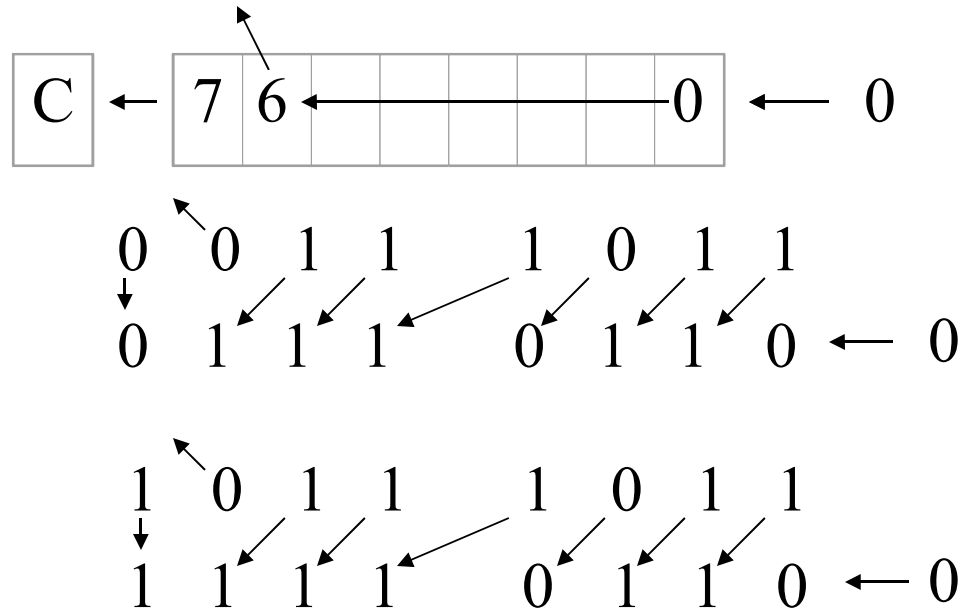
Aritmetický posun obsahu pamäťového miesta **doprava**



ale pre bajt



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



ale pre bajt

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

$$\begin{array}{c}
 \overbrace{\hspace{10em}} \\
 \downarrow \\
 (0 \ 0 \ 1 \ 1 \quad 1 \ 0 \ 1 \ 1)_2 = 32 + 16 + 8 + 2 + 1 = (59)_{10} \\
 (0 \ 0 \ 0 \ 1 \quad 1 \ 1 \ 0 \ 1)_2 = 16 + 8 + 4 + 1 = (29)_{10}
 \end{array}$$

$$59 : 2 = 29 \quad R = 1$$

$$a = b \gg n;$$

- celočíselný posun **dol'ava** je násobením číslom 2
 pri n posunoch ide o násobenie číslom 2^n

$$\begin{array}{l}
 (0 \ 0 \ 1 \ 1 \quad 1 \ 0 \ 1 \ 1)_2 = 32 + 16 + 8 + 2 + 1 = (59)_{10} \\
 (0 \ 1 \ 1 \ 1 \quad 0 \ 1 \ 1 \ 0)_2 = 64 + 32 + 16 + 4 + 2 = (118)_{10} \\
 59 * 2 = 118
 \end{array}$$

$$a = b \ll n;$$

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 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 čísel bez znamienka

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

$$S = A + B$$

<i>A</i>	<i>B</i>	<i>S</i>	<i>carry (C)</i>
0	0	0	0
1	0	1	0
0	1	1	0
1	1	0	1

V prípade súčtu dvoch 8-bitových slov bude

	0	0	1	1	1	0	1	1	$(59)_{10}$	
	+	0	1	1	1	0	1	1	0	$(118)_{10}$
<i>carry</i> $\langle 0 \rangle$	1	1	1	1	1	1	0	1		
=	1	0	1	1	0	0	0		$(177)_{10}$	
$59 + 118 = 177$										

v ďalšom príklade

	1	0	1	1	1	0	1	1	$(187)_{10}$	
	+	0	1	1	1	0	1	1	0	$(118)_{10}$
<i>carry</i> $\langle 1 \rangle$	1	1	1	1	1	1	0	0		
=	0	0	1	1	0	0	0	1	$(49)_{10}$	
$187 + 118 = 305$										

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 vieme, pamäťové miesto nestačí svojou veľkosťou na zápis výsledku sčítania.

- pri viacslovnom sčítaní musíme uvažovať hodnotu tohoto 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 platí (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

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

Polovičná sčítačka sa dá realizovať pomocou kombinačných obvodov

[1]str.170 Obr. 4.12

Tri možné spôsoby realizácie polovičnej sčítačky.

Pri sčítaní na ďalších radoch treba uvažovať prenosy s predchádzajúcich rádo

$$S = A + B$$

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

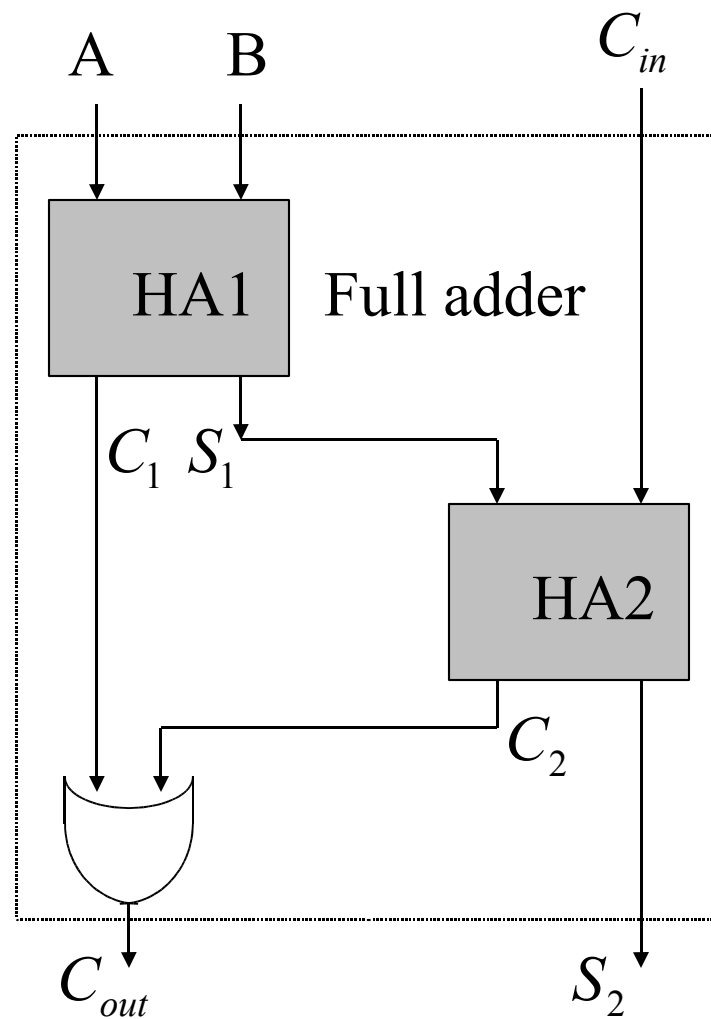
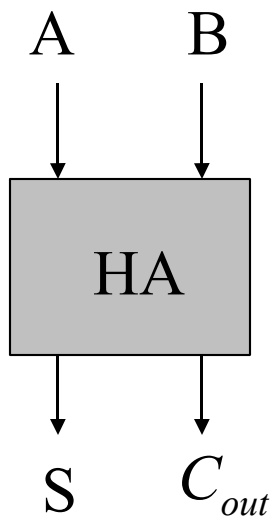
$$s_i = (a_i + b_i + c_i) \bmod 2 \quad c_{i+1} = (a_i + b_i + c_i) / 2$$

$$s_i = \overline{a_i} \overline{b_i} c_i + \overline{a_i} b_i \overline{c_i} + \overline{a_i} \overline{b_i} c_i + a_i b_i c_i \quad c_{i+1} = a_i . b_i + (a_i + b_i) c_i$$

Možné realizácie úplnej sčítačky :

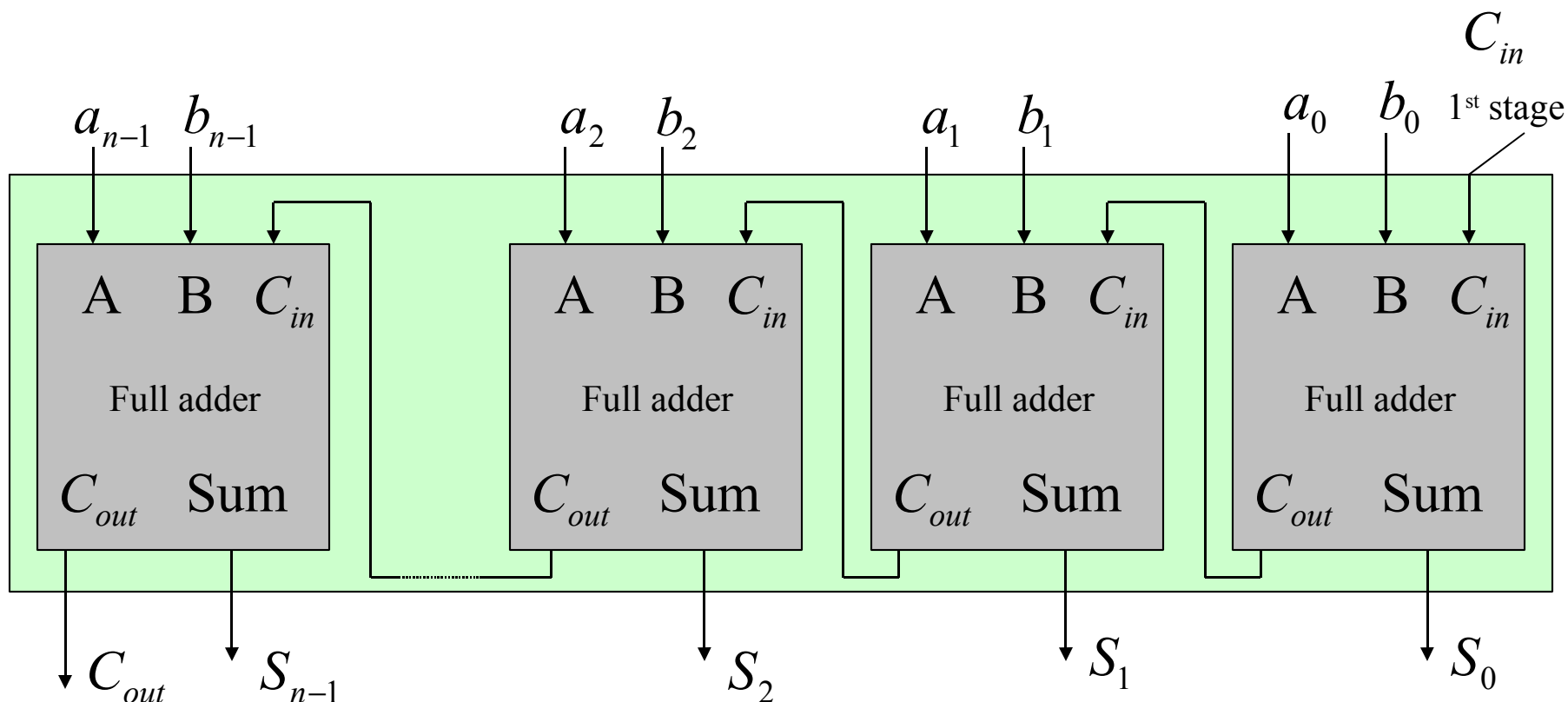
[1]str.173 Obr. 4.18

použitím polovičnej sčítacky :

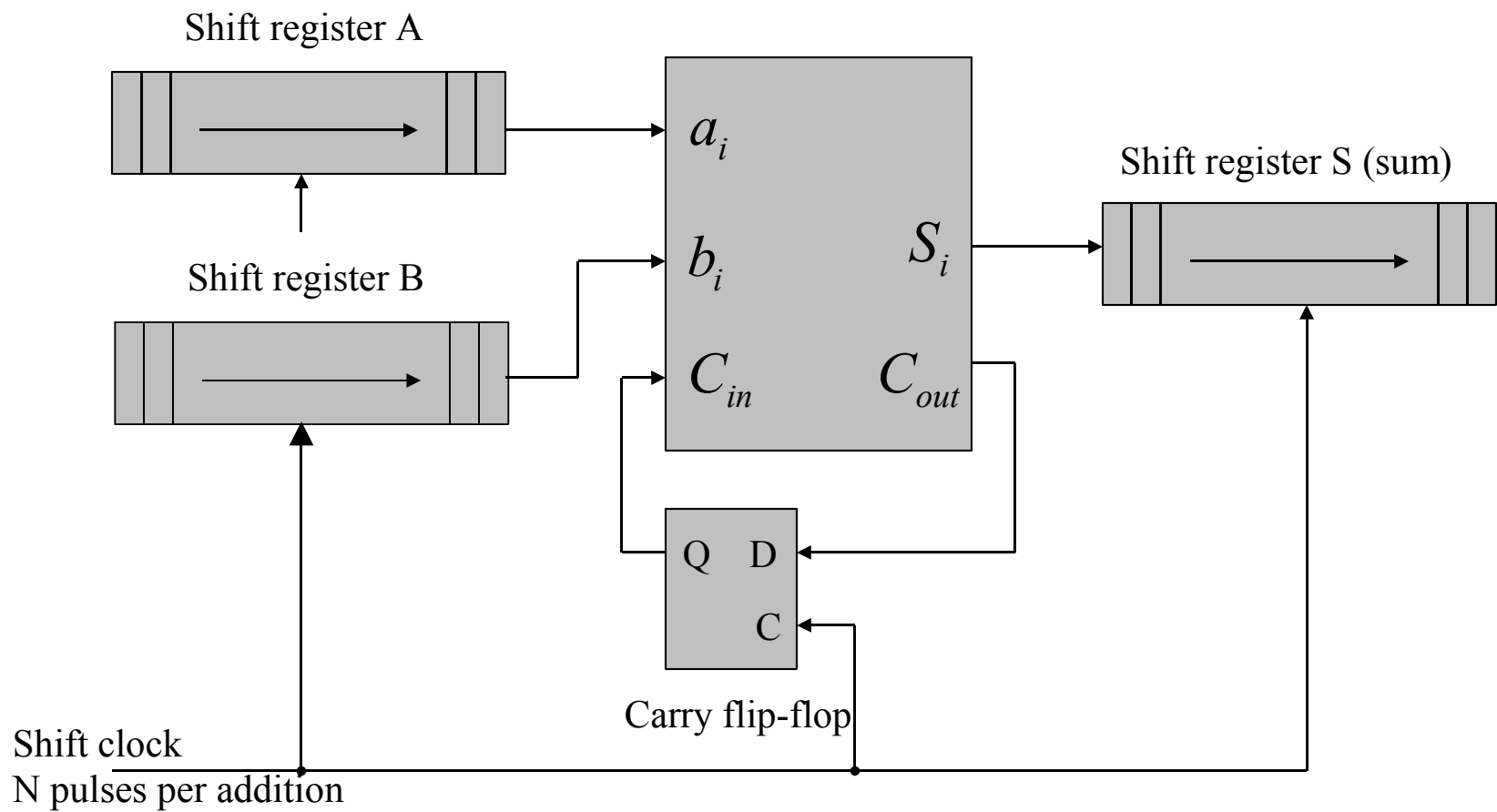


Na sčítanie dvoch n -bitových čísel sa dajú použiť **paralelná** alebo **sériová** sčítačka, pričom sa vo všetkých stupňoch používa úplná sčítačka (realizácia prenosu z predchádzajúceho sčítania)

Inkrementácia: nastavíme carry bit a pričítame nuly



Paralelná sčítačka



Sériová sčítačka

Rozdiel čísel bez znamienka

Pre odčítanie binárnych čísel platia pravidlá binárnej aritmetiky pre jednobitové slová

$$S = A - B$$

<i>A</i>	<i>B</i>	<i>S</i>	<i>borrow (B)</i>
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

$$\begin{array}{r}
 _{10} \\
 - _{10} \\
 \hline
 borrow \langle 0 \rangle _{10} \\
 _{10} \\
 _{10}
 \end{array}$$

$$\begin{array}{r}
 _{10} \\
 - _{10} \\
 \hline
 borrow \langle 1 \rangle _{10} \\
 _{10} \\
 _{10}
 \end{array}$$

$$59 - 181 = -122 \quad [(256 + 59) - 118 = 197]$$

V druhom prípade je po vykonaní operácie nastavený príznak podtečenia

- v prípade viacbajtového odčítania je potrebné tento príznak uvažovať
- inak výsledok je 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

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

MSB = 0 kladné číslo $((-1)^0 = +1)$

MSB = 1 záporné číslo 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 $(-0)_{10}$

Inverzný kód (1'st complement)

Bit s najväčšou váhou MSB – znamienkový bit

MSB = 0 kladné číslo

MSB = 1 záporné číslo

Číslo s opačným znamienkom sa vytvára inverziou bit po bite

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

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

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

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

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

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

$$\begin{array}{r} -4 \ 1111 \ 1011 \\ \hline \end{array} \quad \begin{array}{r} 4 \ 1 \ 0000 \ 0011 \\ \hline \end{array}$$

$$\begin{array}{r} -4 \ 1111 \ 1011 \\ \hline \end{array} \quad \begin{array}{r} 4 \ 1 \ 0000 \ 0011 \\ \hline \end{array}$$

$$\begin{array}{r} \mapsto \quad \quad \quad 1 \\ \hline 0000 \ 0100 \end{array}$$

pri sčítaní treba korigovať výsledok pripočítaním obsahu carry bitu k LSB

- nejednoznačná nula

$$\begin{array}{ccccccccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & (0)_{10} \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & (-0)_{10} \end{array}$$

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

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

Doplnkový kód (2'st 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

- inverzia bit po bite

- pripočítanie jednotky (inkrementácia)

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

- 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
<u>+5 0000 0101</u>	<u>−5 1111 1011</u>
−4 1111 1100	4 $\langle 1 \rangle$ 0000 0100

- v prípade súčtu čísel v doplnkovom kóde nastavenie carry bitu neznamena 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!

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

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

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

$$\begin{array}{rcl}
 & \boxed{00} & 0000110 \quad (+6) \\
 + & \boxed{00} & 0001000 \quad (+8) \\
 \hline
 & & 00001110 \quad (+14)
 \end{array}$$

V=0, C=0 => OK

$$\begin{array}{rcl}
 & \boxed{01} & 111111 \quad (+127) \\
 + & \boxed{00} & 000001 \quad (+1) \\
 \hline
 & & 10000000 \quad (-128)
 \end{array}$$

V=1, C=0 => ERROR

$$\begin{array}{rcl}
 & \boxed{00} & 000010 \quad (+2) \\
 + & \boxed{11} & 111100 \quad (-4) \\
 \hline
 & & 1111110 \quad (-2)
 \end{array}$$

V=0, C=0 => OK

$$\begin{array}{rcl}
 & \boxed{00} & 000100 \quad (+4) \\
 + & \boxed{11} & 111110 \quad (-2) \\
 \hline
 (C) & & 00000010 \quad (+2)
 \end{array}$$

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

$$\begin{array}{r}
 \begin{array}{|c|c|} \hline 1 & 1 \\ \hline \end{array} 1\ 1\ 1\ 1\ 1\ 0 & (-2) \\
 + \begin{array}{|c|c|} \hline 1 & 1 \\ \hline \end{array} 1\ 1\ 1\ 1\ 0\ 0 & (-4) \\
 \hline
 \end{array}$$

$$(C)1\ 1\ 1\ 1\ 1\ 0\ 1\ 0 \quad (-6)$$

V=0, C=1 => OK

$$\begin{array}{r}
 \begin{array}{|c|c|} \hline 1 & 0 \\ \hline \end{array} 0\ 0\ 0\ 0\ 0\ 1 & (-127) \\
 + \begin{array}{|c|c|} \hline 1 & 1 \\ \hline \end{array} 0\ 0\ 0\ 0\ 1\ 0 & (-62) \\
 \hline
 \end{array}$$

$$(C)0\ 1\ 0\ 0\ 0\ 0\ 1\ 1 \quad (+67)$$

V=1, C=1 => ERROR

BCD kód

BCD (Binary Coded Decimal) – dvojkovo-desiatkové kódovanie

- zostáva desiatková sústava
- číslce sa kódujú do binárneho kódu

Existuje veľa možností kódovať dekadické číslce. 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

$$d_i = a_3 a_2 a_1 a_0 \quad \Rightarrow \quad d_j = \overline{a_3} \overline{a_2} \overline{a_1} \overline{a_0}$$

4. Usporiadanosť – väčším dekadickým číslom sú priradené väčšie kódy (možnosť porovnávania čísl)
5. Párnosť – párnym číslom zodpovedajú párne kódy, nepárnym nepárne

Najpoužívanější kód je pre :

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

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

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

Pozn.: Môžeme sa stretnúť s pojmami zhustený a nezhusený 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

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

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 $23+45$, $23+29$, $27+29$

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

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

$$\begin{array}{rcccl}
 & 23 & 0010 & 0011 & \\
 + & 29 & 0010 & 1001 & \\
 \hline
 & & 0 & & < AC > \\
 & 52 & 0100 & 1100 & [4C]_H
 \end{array}$$

$$\begin{array}{rcccl}
 & 27 & 0010 & 0111 & \\
 + & 29 & 0010 & 1001 & \\
 \hline
 & & 1 & & < AC > \\
 & 56 & 0101 & 0000 & [50]_H
 \end{array}$$

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) .

Po vykonaní súčtu BCD čísel treba testovať

- buď prenos do vyššieho rádu z každej štvorice bitov
- alebo, či kód číslice patrí 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

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

0 $\langle AC \rangle$

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

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

1 $\langle AC \rangle$

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

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 :

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

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

[1]str.196 Obr. 4.31

```
unsigned long sucin(unsigned short a,unsigned short b)
{
    unsigned long result = 0, mult = a;
    int i;
    while(b){
        if (b & 1) result += mult;
        mult <<= 1; b >>= 1;
    }
    return result;
}
```

Realizácia násobenia v jazyku C

[2]str.213 Obr. 7-6

Blokov schéma celočíselnej násobičky

Súčin celých čísel (so znamienkom)

Budeme uvažovať len čísla vyjadrené v doplnkovom kóde. Na to aby algoritmus násobenia bol použiteľný, okrem možnosti previesť operandy na kladné čísla, a potom upraviť výsledok podľa výsledného znamienka súčinu, je potrebné 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.

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

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.

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) :

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

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

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

$$15 * 13 = 195$$

$$C \quad 9 \quad 8 \quad 7 \quad 6 \quad 5 \quad 4 \quad 3 \quad 2 \quad 1 \quad 0$$

$$0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad \underline{1 \quad 0^*} \quad (-)$$

0 1 0 0 0 1 0 0 0 0 0

$$0 \ 1 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \quad (\rightarrow)$$

$$0 \ 1 \ 1 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \quad 0 \ 1 \ 1 \ \underline{0} \ 1 \quad (+)$$

0 0 1 1 1 1 0 0 0 0 0

$$1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad (\rightarrow)$$

$$0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad \underline{1 \quad 0} \quad 1 \quad (-)$$

0 1 0 0 0 1 0 0 0 0 0

$$0 \quad 1 \quad 0 \quad 1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad (\rightarrow)$$

$$0 \ 1 \ 1 \ 0 \ 1 \ 0 \ 0 \ 1 \ 1 \ 0 \ 0 \quad 0 \ \underline{1 \ 1} \ 0 \ 1 \quad (\rightarrow)$$

$$0 \ 1 \ 1 \ 1 \ 0 \ 1 \ 0 \ 0 \ 1 \ 1 \ 0 \quad 0 \ 1 \ 1 \ 0 \ 1 \quad (+)$$

0 0 1 1 1 1 0 0 0 0 0

$$1 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad (\rightarrow)$$

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

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

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

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

C 9 8 7 6 5 4 3 2 1 0

0 0 0 0 0 0 0 0 0 0 0

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

0 1 1 1 0 0 0 1 0 0 0

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

0 0 0 0 1 0 1 1 0 1 0

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

1 0 0 1 1 0* (-)

($\rightarrow$)

1 0 0 1 1 ($\rightarrow$)

1 0 0 1 1 (+)

($\rightarrow$)

1 0 0 1 1 ($\rightarrow$)

1 0 0 1 1 (-)

($\rightarrow$)

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

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 medizvysledkom násobenia, stačí jeden súčet a jeden rozdiel posunutého operandu A.

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

Urýchlenie násobenia

- **vyhl'adávacia tabuľka** (look-up table) adresa do tabuľky dvojica operandov, obsah tabuľky výsledok súčin operandov

problémom je veľkosť tabuľky – operandy 8-bitové $\Rightarrow$ adresa 16-bitová $\Rightarrow 2^{16}=65536$ výsledkov, výsledky sú 16-bitové $\Rightarrow$ potrebná pamäť 128 kB

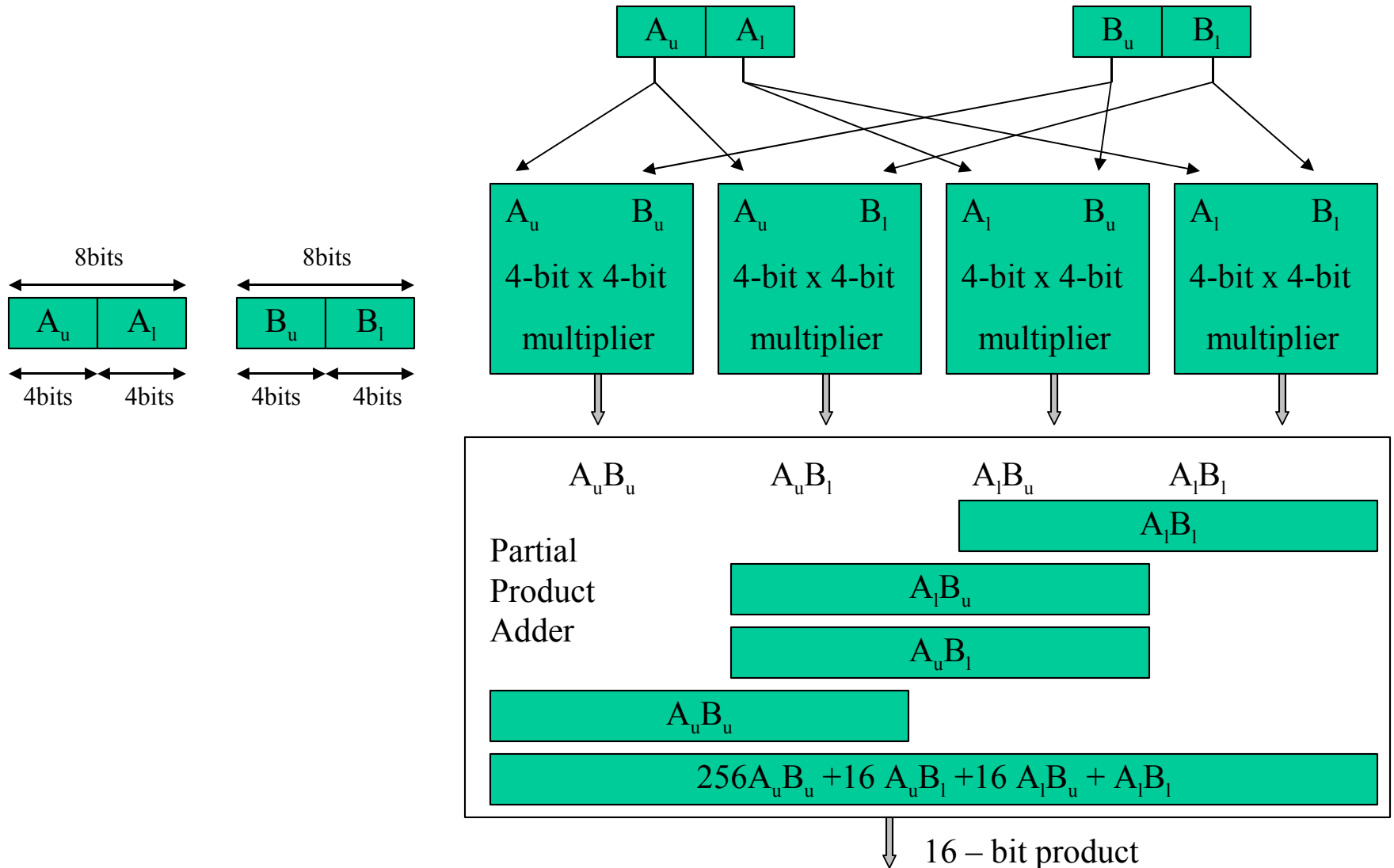
- modifikáciou tejto myšlienky je rozdelenie operandov na menšie časti

$$\begin{aligned} 34 \times 27 &= (3 \times 10 + 4)(2 \times 10 + 7) \\ &= 3 \times 2 \times 10^2 + 3 \times 7 \times 10 + 4 \times 2 \times 10 + 4 \times 7 \\ &= 6 \times 10^2 + 21 \times 10 + 8 \times 10 + 28 \\ &= 6 \times 10^2 + 29 \times 10 + 28 \\ &= 600 + 290 + 28 \\ &= 918 \end{aligned}$$

$$A = A_u \times 16 + A_l \text{ and } B = B_u \times 16 + B_l$$

$$\text{Consequently, } A \times B = (A_u \times 16 + A_l)(B_u \times 16 + B_l)$$

$$= 256A_uB_u + 16A_uB_l + 16A_lB_u + A_lB_l$$



- alebo použitie hardvérovej násobičky realizovanej hradlovým polom (kombinačný obvod)

Podiel celých čísel bez znamienka

Pre celočíselné delenie platí

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

$$575 : 25 = 23$$

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

$$75$$

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

$$0$$

$$575 : 45 = 12$$

$$\begin{array}{r} - \underline{45} \end{array}$$

$$125$$

$$\begin{array}{r} - \underline{90} \end{array}$$

$$35$$

$$575 = 100011111_2 \quad 25 = 011001_2 \quad -25 = 100111_2$$

$$575 : 25 = 23$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Bloková schéma :

[1]str.202 Obr. 4.36

Urýchlenie algoritmu :

po odčítaní deliteľa a zápornom medzivýsledku
nepripočítat' naspäť delenca, ale pripočítat' $\text{delenec}/2$,
pretože

$$-\text{delenec} + \text{delenec}/2 = -\text{delenec}/2$$

čo je ekvivalentné pripočítaniu deliteľa naspäť
a následnému odčítaniu polovice deliteľa.

```

short podiel(long delenec, short delitel)
{
    short result = 0;
    long del = delitel;
    int i = 0;
    if (delenec < 0 || delitel <= 0) return -1; // chyba
    // zarovnanie delitela dolava
    while(!(del & 0x4000)){del <<= 1; i++;}
    while(i--){
        result <<= 1;
        delenec -= del;
        if (delenec < 0){delenec += del;}
        else {result++;}
        del >>= 1;
    }
    return result;
}

```

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 ...,