

# Компютърни архитектури CSCB008

---

Системи за двоично кодиране. Математика с двоични  
числа

доц. д-р Ясен Горбунов  
2021

Ракетата Ариана 5 изпраща два 3-тонни сателита в космоса..., 4 юни 1996



<https://hackaday.com/2016/06/30/fail-of-the-week-in-1996-the-7-billion-dollar-overflow/>

## Грешка за 7 милиарда долара...



Ариана 5 е двустепенна  
тежкотоварна ракета-носител

дължина — 53 m

диаметър — 5.4 m

тегло – 780 тона

- Унищожената ракета и нейният товар са на стойност 500 милиона USD.
- Разработката ѝ отнема 10 години и струва 7 милиарда USD.

### Причината:

64-bit число с плаваща запетая, свързано с хоризонталната скорост на ракетата е конвертирано в 16-bit цяло число със знак. Размерът на числото надхвърля 32767, което е най-голямото цяло число в този формат, поради което настъпва препълване на разрядната решетка.



## Събиране на числа

десетично събиране

$$\begin{array}{r} 11 \leftarrow \text{пренос} \\ 3734 \\ + 5168 \\ \hline 8902 \end{array}$$

## Събиране на числа

десетично събиране

$$\begin{array}{r} 11 \leftarrow \text{пренос} \\ 3734 \\ + 5168 \\ \hline 8902 \end{array}$$

двоично събиране

$$\begin{array}{r} 11 \leftarrow \text{пренос} \\ 1011 \\ + 0011 \\ \hline 1110 \end{array}$$

двоично събиране с **препълване**

$$\begin{array}{r} 111 \leftarrow \text{пренос} \\ 1011 \\ + 0110 \\ \hline 10001 \end{array}$$

Преносът никога няма да бъде по-голям от единица!

# Математика с двоични числа

Четири възможни резултата от събиране на два бита (два операнда)

$$\begin{array}{r} 0 \\ + 0 \\ \hline 0 \end{array} \quad \begin{array}{r} 0 \\ + 1 \\ \hline 1 \end{array} \quad \begin{array}{r} 1 \\ + 0 \\ \hline 1 \end{array} \quad \begin{array}{r} 1 \\ + 1 \\ \hline 10 \end{array} \quad \begin{array}{c} 1 \\ \leftarrow \text{пренос} \end{array}$$

Четири възможни резултата от събиране на три бита (два операнда и пренос)

$$\begin{array}{r} 1 \\ 0 \\ + 0 \\ \hline 1 \end{array} \quad \begin{array}{r} 1 \\ 1 \\ 0 \\ + 1 \\ \hline 10 \end{array} \quad \begin{array}{r} 1 \\ 1 \\ 1 \\ + 0 \\ \hline 10 \end{array} \quad \begin{array}{r} 1 \\ 1 \\ 1 \\ + 1 \\ \hline 11 \end{array} \quad \begin{array}{c} 1 \\ \leftarrow \text{пренос} \end{array}$$

Събиране на два байта

$$\begin{array}{ccccccccc} & & & & 1 & 1 & 1 & 1 & 1 \\ & & & & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ + & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ \hline 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}$$

## Получаване на числа в BCD код

### алгоритъм „Double dabble“ (shift-and-add-3)

За преобразуване на  $n$ -битово число:

1. Заделят се  $n + 4 \cdot \text{ceil}(\frac{n}{3})$  бита; двоичният оригинал се намира отдясно  
(„ceil“ закръгля до най-близкото цяло число, което е по-голямо или равно на аргумента)
2. Проверява се дали някоя BCD цифра е  $>4$ ; ако е, то се прибавя  $3_{(10)} = 0011_{(2)}$
3. Извършва се преместване наляво
4. След общо  $n$  премествания алгоритъмът приключва



## Получаване на числа в BCD код

### алгоритъм „Double dabble“ (shift-and-add-3)

#### Пример

За числото 11110011 са нужни

$$n + 4 \cdot \text{ceil}\left(\frac{n}{3}\right) = 8 + 4 \cdot \text{ceil}\left(\frac{8}{3}\right) = 8 + 4 \cdot 3 = 20$$

бита

стотици	десетици	единици		
0000	0000	0000	<b>11110011</b>	инициализация
0000	0000	0001	11100110	преместване (<<)
0000	0000	0011	11001100	преместване (<<)
0000	0000	0111	10011000	преместване (<<) $(111_{(2)} = 7_{(10)} > 4_{(10)})$
0000	0000	1010	10011000	+ 3 към единиците
0000	0001	0101	00111000	преместване (<<) $(101_{(2)} = 5_{(10)} > 4_{(10)})$
0000	0001	1000	00111000	+ 3 към единиците
0000	0011	0000	01100000	преместване (<<)
0000	0110	0000	11000000	преместване (<<) $(110_{(2)} = 6_{(10)} > 4_{(10)})$
0000	1001	0000	11000000	+ 3 към десетиците
0001	0010	0001	10000000	преместване (<<)
0010	0100	0011	00000000	преместване (<<)
<b>2</b>	<b>4</b>	<b>3</b>		<b>BCD</b>



## Събиране на числа в BCD код

при BCD събиране сборът не може да превишава  $9 + 9 + 1 = 19$

$9 + 9 + 1 = 19$   
BCD      BCD      пренос

ДВОИЧНО

$$\begin{array}{r} 9_{\text{DEC}} \quad 1001 \\ + 9_{\text{DEC}} \quad 1001 \\ + 1_{\text{DEC}} \quad 0001 \\ \hline 19_{\text{DEC}} \quad 10011 \end{array}$$



в BCD код

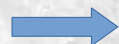
$$\begin{array}{r} 9_{\text{DEC}} \quad 1001 \\ + 9_{\text{DEC}} \quad 1001 \\ + 1_{\text{DEC}} \quad 0001 \\ \hline 0001 \quad 0011 \\ \underline{\quad 1 \quad} \quad \underline{\quad 3 \quad} \end{array}$$

Невярно в BCD код!

## Събиране на числа в BCD код

ДВОИЧНО

$$\begin{array}{r}
 9_{\text{DEC}} \quad 1001 \\
 + 9_{\text{DEC}} \quad 1001 \\
 + 1_{\text{DEC}} \quad 0001 \\
 \hline
 19_{\text{DEC}} \quad 10011
 \end{array}$$



BCD код

$$\begin{array}{r}
 9_{\text{DEC}} \quad 1001 \\
 + 9_{\text{DEC}} \quad 1001 \\
 + 1_{\text{DEC}} \quad 0001 \\
 \hline
 0001 \quad 0011
 \end{array}$$

1

3

→ грешен BCD

Ако сумата е  $\geq 1010_{\text{BIN}}$  ( $1001_{\text{BIN}}$  + пренос), трябва да се добави  $6_{\text{DEC}} = 0110$

BCD код

$$\begin{array}{r}
 9_{\text{DEC}} \quad 1001 \\
 + 1_{\text{DEC}} \quad 0001 \\
 \hline
 1010 > 1001
 \end{array}$$



1) add 6

$$\begin{array}{r}
 1010 \\
 + 0110 \\
 \hline
 1 \quad 0000
 \end{array}$$

2)

$$\begin{array}{r}
 1 \quad 0000 \\
 + 1001 \\
 \hline
 0001 \quad 1001
 \end{array}$$

1

9

→ правилен BCD

## Събиране на числа в BCD код

### Примери

Десетично събиране  
и преобразуване в BCD код

	8 <sub>DEC</sub>	1000
+	9 <sub>DEC</sub>	1001
<hr/>		
17 <sub>DEC</sub>		10001
+	6 <sub>DEC</sub>	0110
<hr/>		
	0001	0111
	<u>1</u> <sub>BCD</sub>	<u>7</u> <sub>BCD</sub>

*Note: A blue arrow points from the '1' in 17<sub>DEC</sub> to the '1' in 1<sub>BCD</sub>.*

Събиране в BCD код

	184 <sub>BCD</sub>	0001	1000	0100
+	576 <sub>BCD</sub>	0101	0111	0110
<hr/>				
		0110	1111	1010
	+	6 <sub>DEC</sub>	0000	0110
<hr/>				
760 <sub>BCD</sub>		0111	10110	10000
		<u>7</u> <sub>BCD</sub>	<u>6</u> <sub>BCD</sub>	<u>0</u> <sub>BCD</sub>

*Note: Blue arrows point from the circled '1' in 10110 to the '6' in 6<sub>BCD</sub>, and from the circled '1' in 10000 to the '0' in 0<sub>BCD</sub>.*



## Изваждане на двоични числа в BCD код

Въвеждане на код за знак:

за положителни числа  $\rightarrow$  код  $0000_{\text{BIN}} = 0_{\text{DEC}}$

за отрицателни числа  $\rightarrow$  код  $1001_{\text{BIN}} = 9_{\text{DEC}}$

**Метод с допълнение на 9**

Числото, получено чрез изваждане на число от основата на бройната система. Например, десетичното допълнение на 8 е 2.

Допълненията се използват в цифровите схеми, защото изваждането става по-бързо чрез използване на допълнения, отколкото чрез извършване на истинско изваждане.

1. Извършва се допълнение на 9 на умалителя
2. Събират се двете числа
3. При необходимост се извършва корекция ( $+6_{\text{DEC}} = 0110_{\text{BIN}}$ )
4. Ако **възникне пренос**, то резултатът е **положителен** и вместо пренасяне се извършва сумиране с бита за пренос (1)
5. Ако **не възникне пренос**, то резултатът е **отрицателен** и се извършва ново допълнение на 9
6. Представят се числата с добавен знак (0000 (+) или 1001 (-))

# Математика с двоични числа

## Изваждане на двоични числа в BCD код

Пример:  $357 - 432 = -75$

3 5 7  
 $357_{\text{DEC}} = 0011\ 0101\ 0111_{\text{BCD}}$

допълнение на 9  
 $-432_{\text{DEC}} = 999 - 432 = 567$

5 6 7  
 $0101\ 0110\ 0111_{\text{BCD}}$

1

$0011\ 0101\ 0111$   
+  $0101\ 0110\ 0111$

=  $1000\ 1011\ 1110$   
8 11 14

невалидно

2

$1000\ 1011\ 1110$   
8 11 14

+  $0000\ 0110\ 0110$   
0 6 6

=  $1001\ 0010\ 0100$   
9 2 4

корекция

3

не възниква пренос – **отрицателен**  
резултат – допълнение на 9

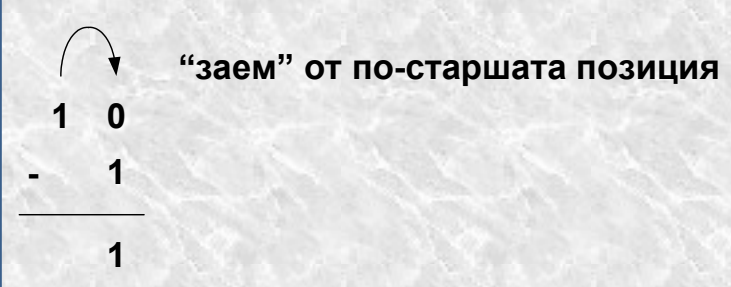
$999 - 924 = 075$

$-75_{\text{DEC}} = 1001\ 0111\ 0101_{\text{BCD}}$

4

## Двоично изваждане

$$\begin{array}{r} \phantom{0} 0 \phantom{0} 1 \phantom{0} 0 \phantom{0} 0 \phantom{0} 0 \phantom{0} 0 \phantom{0} 0 \phantom{0} 0 \\ \phantom{0} 1 \phantom{0} 1 \phantom{0} 1 \phantom{0} 1 \phantom{0} 1 \phantom{0} 1 \phantom{0} 1 \phantom{0} 1 \phantom{0} 1 \\ - \phantom{0} 0 \phantom{0} 0 \phantom{0} 1 \phantom{0} 0 \phantom{0} 1 \phantom{0} 0 \phantom{0} 1 \phantom{0} 0 \phantom{0} 1 \\ \hline 0 \phantom{0} 1 \phantom{0} 1 \phantom{0} 1 \phantom{0} 0 \phantom{0} 0 \phantom{0} 1 \phantom{0} 1 \phantom{0} 0 \end{array}$$



“заем” от по-старшата позиция

подобно на десетичното изваждане

## Проблеми

- нужен е начин за представяне на знака (+/-) в двоична бройна система {0; 1}
- синтез на хардуерно изваждащо устройство е възможен, но излишен

## Решение

Изваждането е частен случай на събирането!



**Представяне на двоични числа със знак**

**два основни метода**

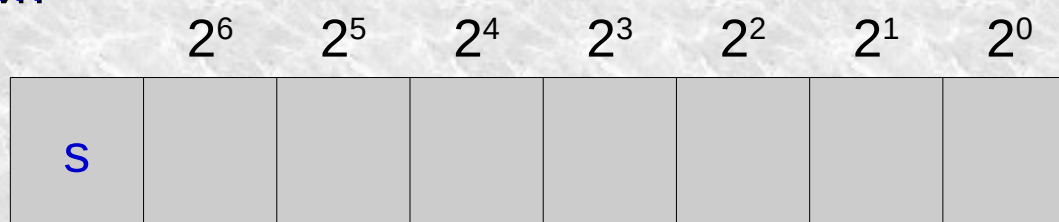
**Величина със знак  
(Sign / Magnitude)**

**Допълнителен код  
(Two`s Complement)**

## Представяне на двоични числа със знак

### Величина със знак (Sign / Magnitude)

жертване на един бит



$s=0$  +

$s=1$  -

представяне

$$A: \{a_{N-1}, a_{N-2}, \dots, a_2, a_1, a_0\}$$

$$A = (-1)^{a_{N-1}} \sum_{i=0}^{N-2} a_i 2^i$$

← за положителните числа →

обхват

$$[0, 2^{N-1}-1]$$

← за отрицателните числа →

$$[-2^{N-1}-1, 2^{N-1}-1]$$

## Представяне на двоични числа със знаков бит (sign / magnitude)

Пример

$$+6_{\text{DEC}} = 0\ 1\ 1\ 0$$

$$-6_{\text{DEC}} = 1\ 1\ 1\ 0$$

сумирането не работи

$$\begin{array}{rcl} & +6 & 0\ 1\ 1\ 0 \\ + & -6 & 1\ 1\ 1\ 0 \\ \hline & & 1\ 0\ 1\ 0\ 0 \end{array}$$

грешно

двусмислие в представянето на 0

$$\begin{array}{rcl} & +0 & 0\ 0\ 0\ 0 \\ & -0 & 1\ 0\ 0\ 0 \end{array}$$



## Представяне на двоични числа със знак

### Двоични допълнения

Всяко число има обратно, чиято стойност добавена към оригиналното дава 0.

## Допълнителен код (Two`s Complement)

### 1. прав код (П.К.)

абсолютната стойност на числото

Пример:  $|+120_{DEC}| = 01111000_{BIN}$  необходим е допълнителен бит за знак  
 $|-115_{DEC}| = 01110011_{BIN}$

### 2. обратен код (О.К.)

допълнение на единицата (One`s Complement)

- при положителни числа О.К. = П.К.
- при отрицателни числа е необходимо инвертиране на всички битове

### 3. допълнителен код (Д.К.)

- при положителни числа Д.К. = П.К.
- при отрицателни числа към числото в О.К. се добавя 1

## Допълнителен код

представяне

$$A: \{a_{N-1}, a_{N-2}, \dots, a_2, a_1, a_0\}$$

$$A = a_{N-1}(-2^{N-1}) \sum_{i=0}^{N-2} a_i 2^i$$

обхват

$$[0, 2^{N-1}-1]$$

← за положителните числа →

← за отрицателните числа →

$$[-2^{N-1}, 2^{N-1}-1]$$

преобразуване в права посока

П.К.

О.К.

Д.К.

преобразуване в обратна посока

Д.К.

О.К.

П.К.

**Допълнителния код на допълнителния е правия код!**

## Допълнителен код – примери

положително число

$$N_{\text{DEC}} = 21$$

прав код  $N_{\text{BIN}} = 010101$

обратен код  $N_{\text{BIN}} = 010101$   
не се извършва преобразуване

отрицателно число

$$N_{\text{DEC}} = -10$$

прав код  $N_{\text{BIN}} = 01010$   
инвертиране

обратен код  $N_{\text{BIN}} = 10101$

допълнителен код  $N_{\text{BIN}} = 10101$   
+ 1  
 $N_{\text{BIN}} = 10110$



## Допълнителен код – примери

### преобразуване

$$N_{\text{DEC}} = 11$$

				LSB
11 : 2	= 5.5	→	1	▲
5 : 2	= 2.5	→	1	↑
2 : 2	= 1	→	0	
1 : 2	= 0.5	→	1	
				MSB

$$N_{\text{BIN}} = 1011$$

$$+ \rightarrow \text{П.К.} = \text{Д.К.} \rightarrow 01011$$

## Допълнителен код – примери

### преобразуване

$$N_{\text{DEC}} = 11$$

				LSB
11	:	2	= 5.5	→ 1
5	:	2	= 2.5	→ 1
2	:	2	= 1	→ 0
1	:	2	= 0.5	→ 1
				MSB

$$N_{\text{BIN}} = 1011$$

$$+ \rightarrow \text{П.К.} = \text{Д.К.} \rightarrow 01011$$

$$N_{\text{DEC}} = -15$$

15	:	2	= 7.5	→ 1
7	:	2	= 3.5	→ 1
3	:	2	= 1.5	→ 1
1	:	2	= 0.5	→ 1

$$N_{\text{BIN}} = (-)1111$$

$$- \rightarrow \text{П.К.} = 01111$$

$$\text{О.К.} = 10000$$

$$+ \quad \quad \quad 1$$

$$\text{Д.К.} = 10001$$

## Допълнителен код – примери

### преобразуване

$$N_{\text{DEC}} = 11$$

11 : 2	= 5.5	→ 1	↑
5 : 2	= 2.5	→ 1	↑
2 : 2	= 1	→ 0	↑
1 : 2	= 0.5	→ 1	↑

LSB

MSB

$$N_{\text{BIN}} = 1011$$

$$+ \rightarrow \text{П.К.} = \text{Д.К.} \rightarrow \mathbf{01011}$$

$$N_{\text{DEC}} = -15$$

15 : 2	= 7.5	→ 1	↑
7 : 2	= 3.5	→ 1	↑
3 : 2	= 1.5	→ 1	↑
1 : 2	= 0.5	→ 1	↑

$$N_{\text{BIN}} = (-)1111$$

$$\begin{array}{r}
 - \rightarrow \text{П.К.} = \mathbf{11111} \\
 \text{О.К.} = \mathbf{10000} \\
 + \quad \quad \quad \mathbf{1} \\
 \hline
 \text{Д.К.} = \mathbf{10001}
 \end{array}$$

$$N_{\text{DEC}} = -73$$

73 : 2	= 36.5	→ 1	↑
36 : 2	= 18	→ 0	↑
18 : 2	= 9	→ 0	↑
9 : 2	= 4.5	→ 1	↑
4 : 2	= 2	→ 0	↑
2 : 2	= 1	→ 0	↑
1 : 2	= 0.5	→ 1	↑

$$N_{\text{BIN}} = (-)1001001$$

$$\begin{array}{r}
 - \rightarrow \text{П.К.} = \mathbf{01001001} \\
 \text{О.К.} = \mathbf{10110110} \\
 + \quad \quad \quad \mathbf{1} \\
 \hline
 \text{Д.К.} = \mathbf{10110111}
 \end{array}$$

## Допълнителен код – примери

### събиране

$$N_{\text{DEC}} = 11 - 15 = -4$$

	11	01011	п.к. = д.к.
+	-15	10001	д.к.
		<u>11100</u>	результат в Д.К.!
		↓ ↓ ↓ ↓ ↓	↓
		00011	О.к.
+		1	↓
		00100	д.к. = п.к.
	(-)	0100 = -4	



## събиране

$$\begin{array}{r}
 11 \quad 01011 \\
 + \quad -15 \quad 10001 \\
 \hline
 111100
 \end{array}$$

П.К. = Д.К.  
 Д.К.  
**результат в Д.К.!**  
 О.К.  
 Д.К. = П.К.

(-)  $0100 = -4$

$$\begin{array}{r}
 -15 \quad 10001 \\
 + \quad 9 \quad 01001 \\
 \hline
 11010 \quad \text{Д.К.} \\
 \downarrow \downarrow \downarrow \downarrow \downarrow \\
 00101 \quad \text{О.К.} \\
 + \quad \quad \quad 1 \\
 00110 \quad \text{Д.К. = П.К.}
 \end{array}$$

(-)  $0110 = -6$

## Допълнителен код – примери

### събиране

$$N_{\text{DEC}} = 11 - 15 = -4$$

	11	01011	п.к. = д.к.
+	-15	10001	д.к.
		11100	результат в Д.К.!
		↓ ↓ ↓ ↓ ↓	
		00011	О.к.
		1	
+		00100	д.к. = п.к.

$$(-) \quad 0100 = -4$$

$$N_{\text{DEC}} = -15 + 9 = -6$$

	-15	10001	
+	9	01001	
		11010	д.к.
		↓ ↓ ↓ ↓ ↓	
		00101	О.к.
		1	
+		00110	д.к. = п.к.

$$(-) \quad 0110 = -6$$

$$N_{\text{DEC}} = 9 - 73 = -64$$

	9	00001001	← водещи нули
+	-73	10110111	
		11000000	д.к.
		↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓	
		00111111	О.к.
		1	
+		01000000 = $-2^6$	д.к. = п.к.

$$(-) \quad 1000000 = -64$$

## Допълнителен код – примери

пет-битов двоичен код на числата от -4 до +4

DEC	4	3	2	1	0	-1	-2	-3	-4
sign magnitude	00100	00011	00010	00001	00000	10001	10010	10011	10100
2`C	00100	00011	00010	00001	00000	11111	11110	11101	11100

+1

Ако към което и да е число, представено в **допълнителен код**, се прибави 1 към най-младшия бит, се получава предхождащото го число. Това позволява по-лесната реализация на аритметичните операции в изчислителните устройства.

## Допълнителен код – бърз начин на получаване

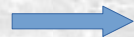
1. Копиране на нулите на числото, започвайки от LSB до достигане на 1.
2. Копиране на достигнатата единица.
3. Ако LSB е 1, то той се копира. Останалите битове се инвертират.

### Пример

П.К.(88<sub>DEC</sub>)      0 1 0 1 1 0 0 0

О.К.                1 0 1 0 0 1 1 1

Д.К.(-88<sub>DEC</sub>)    1 0 1 0 1 0 0 0



1 0 1 0 1 0 0 0

↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓

0 1 0 1 1 0 0 0

първа 1 от  
дясно наляво

Д.К.

П.К.

инвертиране на  
останалите битове

копиране на битовете,  
включително единицата  
от младшия бит



## Сравнение на представянето без знак, със знаков бит и в допълнителен код

Представяне	Обхват	За 1 байт (8-bit)	Брой цели различни числа
Unsigned	$[0, 2^N-1]$	0; 255	256
Sign/Magnitude	$[-2^{N-1}-1, 2^{N-1}-1]$	-127; 127	255
Two's Complement	$[-2^{N-1}, 2^{N-1}-1]$	-128; 127	256

Пример, 4-bit представяне:



Unsigned

0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111

1000 1001 1010 1011 1100 1101 1110 1111 0000 0001 0010 0011 0100 0101 0110 0111

Two's Complement

1111 1110 1101 1100 1011 1010 1001 0000  
1000 0001 0010 0011 0100 0101 0110 0111

Sign/Magnitude

## Увеличаване на битовата разрядност

### Допълване чрез водещ знаков бит

- знаковият бит се копира в най-значещите битове (MSB)
- числото остава същото

Пример 1:

$$\begin{array}{lcl} 3_{4\text{-bit BIN}} & = & 0011 \\ 3_{8\text{-bit BIN}} & = & 00000011 \end{array}$$

Пример 2:

$$\begin{array}{lcl} -5_{4\text{-bit BIN}} & = & 1011 \\ -5_{8\text{-bit BIN}} & = & 11111011 \end{array}$$

### Допълване чрез водещи нули

- копиране на нули в най-значещите битове (MSB)
- отрицателните числа се променят

Пример 1:

$$\begin{array}{lcl} 3_{4\text{-bit BIN}} & = & 0011 \\ 3_{8\text{-bit BIN}} & = & 00000011 \end{array}$$

Пример 2:

$$\begin{array}{lcl} -5_{4\text{-bit BIN}} & = & 1011 \\ -5_{8\text{-bit BIN}} & = & 00001011 \end{array}$$

11<sub>DEC</sub>

## Числа с фиксирана запетая

Не съществува друг начин за определяне на мястото на десетичната точка освен чрез предварителна уговорка.

Пример:  $6.75_{\text{DEC}} = 01101100 = 0110.1100 = 2^2 + 2^1 + 2^{-1} + 2^{-2}$

Числата със знак се представят като „величина със знак“ (Sign / Magnitude) или в допълнителен код.

Пример:  $-2.375$       00100110      абсолютна стойност в регистър

0010.0110      абсолютна стойност ( $2^1 + 2^{-2} + 2^{-3}$ )

1010.0110      величина със знак

виртуална      1101.1010      допълнителен код

## Числа с фиксирана запетая

Събиране/изваждане – допълнителен код

Пример:  $0.75 - 0.625 = 0.125$

$0.75 =$        $0000.1100$       положително число

$-0.625 =$      $0000.1010$        $(2^{-1}+2^{-3})$     П.К.  
                  $1111.0101$       1`s C (О.К.)

+  $0000.0001$   
 $1111.0110$       2`s C (Д.К.)



$0000.1100$	$0.75$
$+$	$+$
$1111.0110$	$(-0.625)$
<hr/>	<hr/>
$10000.0010$	$0.125$

препълване – няма място за този бит



## Числа с плаваща запетая – стандарт IEEE 754

Възможно е представянето на много големи и много малки числа.

Форматът притежава **знак**, **мантиса (M)**, **основа (B)** и **експонента (E)**:  $\pm M \times B^E$

Пример 1:  $342,370,000_{\text{DEC}} = \underbrace{3.4237}_M \times \underbrace{10}_B^{\underbrace{8}_E}$

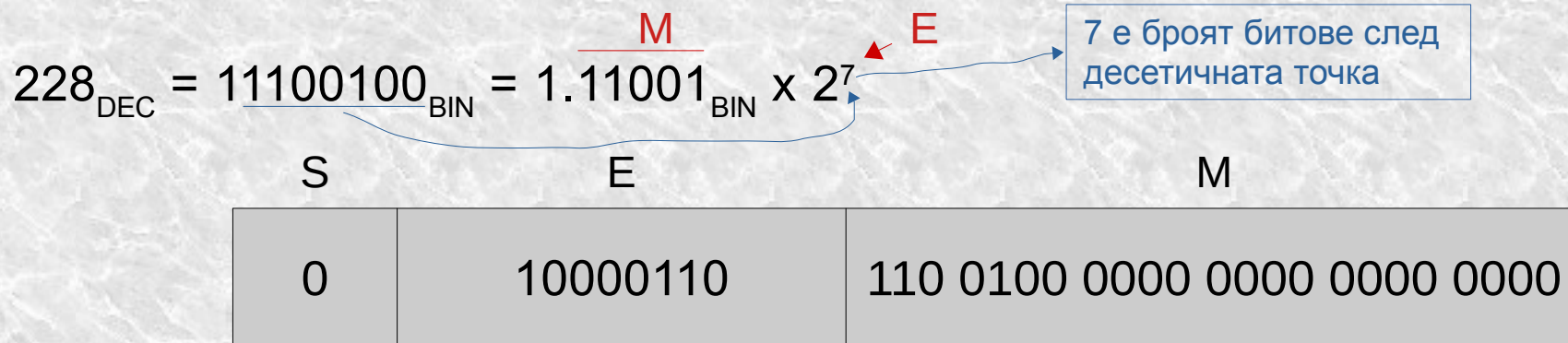
Пример 2:  $100101101.010110_{\text{BIN}} = \underbrace{1.00101101010110}_M \times \underbrace{2}_B^{\underbrace{8}_E}$



## Числа с плаваща запетая – стандарт IEEE 754

### Пример

Да се представи в IEEE 754 числото  $228_{\text{DEC}}$



Експонентата може да бъде положителна или отрицателна. **IEEE 754** използва **отместена експонента**.

$$\pm M \times 2^{(E-127)} \rightarrow E - 127 = 7 \rightarrow E = 7 + 127 = 134_{\text{DEC}} = 10000110_{\text{BIN}}$$

отместване

Първият бит в мантисата е винаги 1 и не се налага да бъде съхраняван.

## Числа с плаваща запетая – стандарт IEEE 754

### Пример

Да се получи 32-bit число според стандарта IEEE 754, представящо  $0.000000110110100101_{\text{BIN}}$ .

$0.000000110110100101$  преместване на десетичната точка със 7 позиции надясно

$$\longrightarrow 0.000000110110100101 = 1.10110100101 \times 2^{-7}$$

знак 0 положително число  
мантиса 10110100101 и 12 нули за допълване до 23 бита  
експонента  $E - 127 = -7$

$$E = -7 + 127 = 120$$

$$120_{\text{DEC}} = 01111000_{\text{BIN}}$$

S E<sub>8-bit</sub> M<sub>23-bit</sub>

0	01111000	101101001010000000000000
---	----------	--------------------------

отговор 00111100010110100101000000000000



## Числа с плаваща запетая – стандарт IEEE 754

- един от първите програмни езици, предлагащи представяне на числа с плаваща запетая е **Fortran** (**Formula Translator**)
- преди IEEE 754-1985, представянето на **double float** числата зависи силно от производителя на компютъра и модела
- най-широко разпространение има **single-precision floating-point**, поради по-широкия обхват спрямо фиксираната запетая при един и същ брой битовете:

32-bit цяло число със знак  $\text{max} = 2^{31} - 1 = 2,147,483,647$

32-bit IEEE 754 floating-point  $\text{max} = (2 - 2^{-23}) \times 2^{127} \approx 3.402823 \times 10^{38}$

Други представяния на числа с плаваща запетая

16-bit: **Half** (binary16)

32-bit: **Single** (binary32), decimal32

64-bit: **Double** (binary64), decimal64

128-bit: **Quadruple** (binary128), decimal128

256-bit: **Octuple** (binary256)

Extended precision formats (40-bit or 80-bit)



## Стандарт IEEE 754

- някои числа не могат да се представят точно – например  $1.7 \approx 1.6999999...$   
за целта калкулатори и финансов софтуер използват BCD код
- поради наличие на водеща единица – проблем при представяне на нула

Представяне на 0, безкрайност и NaN

Число	Знак	Експонента	Мантиса
0	X	0000 0000	0000 0000 0000 0000 0000 000
$\infty$	0	1111 1111	0000 0000 0000 0000 0000 000
$-\infty$	1	1111 1111	0000 0000 0000 0000 0000 000
NaN	X	1111 1111	Non-zero

NaN – Not-a-Number (числа, които не съществуват  $\sqrt{-1}$ ,  $\log_2(-5)$ )

## Стандарт IEEE 754

В процесорите Pentium е съществувал бърк при делене с плаваща запетая (инструкция FDIV) поради неправилно предаване на стойности от LookUp таблица. Това е струвало на Intel 475 милиона долара!

Специален алгоритъм за делене е бил реализиран чрез програмируема логика с 2048 клетки, от които 1066 клетки е трябвало да съдържат една от 5 стойности: -2, -1, 0, +1, +2. В проблемния процесор 5 от клетките, съдържащи +2, са липсвали, в резултат на което е връщана стойност 0.

Tom R. Halfhill (1995). "An error in a lookup table created the infamous bug in Intel's latest processor". BYTE Magazine (March 1995)

$$\frac{4,195,835}{3,145,727} = 1.333820449136241002$$

При преобразуване в HEX стойност на числото  $4,195,835 = 0x4005FB$  цифрата 5 е предизвиквала грешка в логиката за управление на FPU. В резултат, стойността, връщана от процесора, е била погрешна в определени ситуации.

$$\frac{4,195,835}{3,145,727} = 1.333739068902037589$$



66 MHz Intel Pentium

# Математика с двоични числа

## Умножение и деление на степен на две

Операциите умножение и деление са бавни операции!

Най-удобно е умножение и деление на  $2^n$ .

умножение  $\ll 1$  (left shift)

$2^7$	$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$	
0	0	0	0	0	0	1	1	$3_{\text{DEC}}$
0	0	0	0	0	1	1	0	$6_{\text{DEC}}$
1	1	1	0	0	0	0	0	$224_{\text{DEC}}$
1	1	0	0	0	0	0	0	$192_{\text{DEC}}$

препълване  $\rightarrow 1$

деление  $\gg 1$  (right shift)

0	0	0	0	0	0	1	1	$3_{\text{DEC}}$
0	0	0	0	0	0	0	1	$1_{\text{DEC}}$
1	1	1	1	1	1	1	1	$255_{\text{DEC}}$
0	1	1	1	1	1	1	1	$127_{\text{DEC}}$



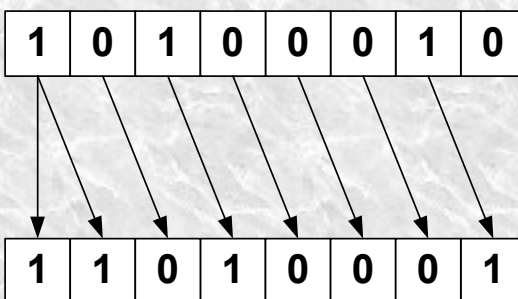
## Умножение и деление на степен на две

### Особености

- излизане извън обхвата – при умножение и препълване
- загуба на точност – при деление и изпадане на бит отдясно

### При числа в допълнителен код

MSB е знаков бит – при изместване на битове надясно трябва да се копира





## Умножение и деление на степен на две

## Умножение, което не е степен на 2

Пример:  $3_{\text{DEC}} = 11_{\text{BIN}}$  да се умножи по 10

1. разлагане на множителя на числа, които са степени на 2  
( $10 = 8 + 2 = 2^3 + 2^1$ )

			1	1
--	--	--	---	---

$3_{\text{DEC}}$

$\times 10$

		1	1	0
--	--	---	---	---

$6_{\text{DEC}}$

1	1	0	0	0
---	---	---	---	---

$24_{\text{DEC}}$

+

1	1	1	1	0
---	---	---	---	---

$30_{\text{DEC}}$

2. умножение на 3 по 2 (  $\ll 1$  )  $\rightarrow$

3. умножение на 3 по 8 (  $\ll 3$  )  $\rightarrow$

4. събиране на двете числа  $\rightarrow$

## Модифициран код – събиране

DEC	BIN
$3+2 = 5$	0011
+	0010
	0101

DEC	BIN
$7+(-5) = 2$	0111
+	д.к. 1011
	1 0010

знаков бит  
→ отрицателно число

Препълването невинаги може да се пренебрегне.

В компютърните системи често се използва **модифициран допълнителен код**

използват се два бита за знак

00 – положителни числа

11 – отрицателни числа

препълването се отчита чрез получаване  
на комбинации 01 или 10

DEC	BIN
$-13+9 = -4$	11 0011
+	00 1001
	11 1100

DEC	BIN
$7+(-5) = 2$	0 0111
+	д.к. 1 1011
	10 0010

## Модифициран код – събиране и изваждане

DEC	BIN
$13+9 = 22$	$00\ 1101$
$+$	$00\ 1001$
	$01\ 0110$

препълване

препълването позволява  
изместване

след изместване  $\rightarrow 00\ 10110$   
(+10110)

DEC	BIN
$13-9 = 4$	$00\ 1101$
$+$	$11\ 0111$
	$100\ 0100$

$00\ 0100$

DEC	BIN
$-13-9 = -22$	$11\ 0011$
$+$	$11\ 0111$
	$110\ 1010$

препълване

$11\ 01010$   
(-10110)

