

ЛУ1 – Преобразуване между бройни системи за цели и дробни числа; понятие за разрядна мрежа; числа с дясно и с ляво фиксирана запетая; прав, обратен и допълнителни кодове; събиране и изваждане на числа с дясно фиксирана запетая; дефиниция за препълване и начини за неговото откриване.

1.1 Позиционни бройни системи. Преобразуване на числата от една позиционна бройна система в друга.

Бройни системи.

Бройна система се нарича способът за изобразяване на числата с помощта на ограничен брой символи, имащи определени количествени значения. Символите, с които се изобразяват числата, се наричат цифри. Наборът от символи (цифри) се нарича азбука на бройната система. Бройната система съдържа съвкупност от правила и съглашения за представяне на числата посредством цифри.

Бройните системи биват позиционни и непозиционни. В позиционните системи всяка цифра има определено тегло, зависещо от позицията на цифрата в последователността, изобразяваща числото. Позицията на цифрата се нарича порядък (разряд). Броят на цифрите, които съдържа бройната система, се нарича основа на бройната система. Едно число може да бъде представено в различни бройни системи. Във всяка от тях то трябва да се композира по един единствен начин. Съществуват алгоритми (правила) за преобразуване на числата от една бройна система в друга.

ДЕСЕТИЧНА БРОЙНА СИСТЕМА

Това е общоприетата бройна система - тази, в която работим (съкращението ѝ е **DEC** - Decimal /десетичен/). За нейното образуване се използват числата от 0 до 9. Преминването от шестнайсетична, осмична и двоична бройни системи в десетична става посредством следната формула:

$$A_q = a_n q^n + a_{n-1} q^{n-1} + \dots + a_0 q^0 + a_{-1} q^{-1} + \dots + a_m q^{-m}$$

където: **q** е основата (бройната система - 2, 8, 10 или 16); **n** - брой позиции (започват от 0, 1, 2, ...) преди десетичната запетая (ако числото е дробно); **m** - брой позиции (започват от 1, 2, ...) след десетичната запетая (ако числото е дробно); **a** - тегловен коефициент.

За да стане по-ясно казаното до тук, ще разгледам няколко примера.

Примери:

- Числото 56_{10} (чете се числото 56 в десетична бройна система) = $5 \cdot 10^1 + 6 \cdot 10^0 = 56$, където (за този пример) $q=10$; $n=1$ (5 заема нулева позиция, 6 - първа).
- Числото $7F,13_{16}$ (чете се числото 7F,13 в шестнайсетична бройна система) = $7 \cdot 16^1 + F \cdot 16^0 + 1 \cdot 16^{-1} + 3 \cdot 16^{-2} = (127 + 19/256)_{10}$, където $q=16$; $n=1$; и $m=2$.
- Числото $10110,1_2$ (чете се числото 10110,1 в двоична бройна система) = $1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 + 1 \cdot 2^{-1} = 22,5_{10}$ ($q=2$; $n=4$; $m=1$).
- Числото 76_8 (чете се числото 76 в осмична бройна система) = $7 \cdot 8^1 + 6 \cdot 8^0 = 62_{10}$ ($q=8$; $n=1$; $m=0$).

ДВОИЧНА БРОЙНА СИСТЕМА

При тази бройна система числата се получават като поредица от нули и единици - **10010**. Всяка една цифра от това число е един бит, като най-старшият бит стои най-вляво на числото. Обикновено този бит (най-старшия) се взема за знаков бит при извършване на числови операции. За да стане по-ясно това погледнете следния пример:

знаков бит	бит 9	бит 8	бит 7	бит 6	бит 5	бит 4	бит 3	бит 2	бит 1	бит 0
1	0	0	1	1	1	0	1	0	1	0

Двоичната бройна система се означава с **BIN** (Binary - двоен). Превръщането на десетично число в двоично става по следния начин:

Алгоритъм:

- Делим числото на две (основата) и получаваме цяла част и остатък. Продължаваме да делим получената цяла част на основата (на две) до получаване на нулева цяла част;
- Получените остатъци от деленията записваме в обратен ред на получаването им.

Пример:

Да се превърне числото 57_{10} в двоична бройна система.

	ц.	ос.
57 :2= 28		1
28 :2= 14		0
14 :2= 7		0
7 :2= 3		1
3 :2= 1		1
1 :2= 0		1



Записваме резултата обратно на получаването му: $57_{10} = 111001_2$

Ако числото е дробно се спазва следният алгоритъм за дробната част:

- Умножаваме дробната част с основата (2) и получаваме нова дробна част и цяла част. Продължаваме да умножаваме получената дробна част по основата докато получим нова дробна част равна на нула;
- Получените цели части при умножението представляват дробната част на новото число. Записват се в реда, в който са получени.

Пример:

Да се превърне числото $57,25_{10}$ в двоична бройна система.

	др.	ц.
$0,25 * 2 = 0,5$		0
$0,5 * 2 = 0$		1




Резултатът се записва в реда на получаването му: $57,25_{10} = 111001,01_2$

ОСМИЧНА БРОЙНА СИСТЕМА

За образуването на тази бройна система се използват числата от 0 до 7. Съкращението ѝ е **ОСТ** (от Octal - осмичен). Превръщането на число от десетична в осмична бройна система става аналогично на превръщането от десетична в двоична, но за основа се взима числото 8 и вместо да се дели (умножава) на две - се дели (умножава) на 8.


Пример:

Да се превърне числото 57_{10} в осмична бройна система.

$$\begin{array}{r|l} \text{ц.} & \text{ос.} \\ 57 : 8 = & 7 \quad 1 \\ 7 : 8 = & 0 \quad 7 \end{array}$$


Записваме резултата обратно на получаването му: $57_{10} = 71_8$

Да се превърне числото $57,25_{10}$ в осмична бройна система.

$$\begin{array}{r|l} \text{др.} & \text{ц.} \\ 0,25 * 8 = & 0 \quad 2 \end{array}$$


Резултатът се записва в реда на получаването му: $57,25_{10} = 71,2_8$

Превръщане от осмична в двоична бройна система

Превръщането става като всяко едно от числата в осмична бройна система се превърне в триади (триадата е поредица от три бита) в двоично число, като се използва [таблицата](#) в началото на документа. Казаното до тук предполагам, че не Ви е ясно и ще разгледаме следният пример.

Пример:

Да се превърне числото 57 от осмична в двоична бройна система.

Правим следното нещо: от горната [таблица](#) превръщаме числата в триади - $5 \rightarrow 101$; $7 \rightarrow 111$. Забележете, че не се взимат всичките четири бита от числото в двоичен код, а само последните три бита, без най-старшия. Събираме двете части на числото и се получава резултатът: $57_8 = 101111_2$.

Превръщане от двоична в осмична бройна система

Това превръщане става по обратния начин - двоичното число се разделя на триади (от дясно на ляво) и от [таблицата](#) се записват съответните осмични числа. Ако в началото на двоичното число битовете не стигат, за да образуват триада, то се попълват с нули.

Пример:

Да се превърне числото **1011110** от двоична в осмична бройна система.

Разделяме числото на триади:

1 | 011 | 110

Тъй като от пред не стигат числата за образуване на триада, то там се попълват нули:

001 | 011 | 110


Отделната триада се превръща в съответното число в осмичен код, посредством [таблицата](#) за бройните системи в началото на документа. Ето и съответните стойности: **001** \rightarrow **1**; **011** \rightarrow **3** и **110** \rightarrow **6**. Сглобяват се получените цифри и се получава числото в осмична бройна система: **1011110₂=136₈**.

ШЕСТНАДЕСЕТИЧНА БРОЙНА СИСТЕМА

За образуването на тази бройна система се използват числата от 0 до 9 и буквите А, В, С, D, Е и F. Съкращението ѝ е **HEX** (от Hexadecimal - шестнадесетичен). Превръщането на число от десетична в шестнадесетична бройна система става аналогично на превръщането от десетична в двоична, но за основа се взима числото 16 и вместо да се дели (умножава) на две - се дели (умножава) на 16.


Пример:

Да се превърне числото 57_{10} в шестнадесетична бройна система.

$$\begin{array}{r|l} \text{ц.} & \text{ос.} \\ 57 : 16 = & 3 \quad 9 \\ 3 : 16 = & 0 \quad 3 \end{array}$$


Записваме резултата обратно на получаването му: $57_{10} = 39_{16}$

Да се превърне числото $57,25_{10}$ в шестнадесетична бройна система.

$$\begin{array}{r|l} \text{др. ц.} & \\ 0,25 \cdot 16 = & 0 \quad 4 \end{array}$$


Резултатът се записва в реда на получаването му: $57,25_{10} = 39,4_{16}$

Превръщане от шестнадесетична в двоична бройна система

Това превръщане става аналогично на осмично-двоичното преобразуване, само, че тук се използват тетради вместо триади (тетрадата е поредица от четири бита в двоичен код). Значи какво правим: всяка една цифра или буква от шестнадесетичното число се превръщат в тетради в двоично число, като се използва [таблицата](#) в началото на документа. Получените тетради се сглобяват и се получава двоичното число.

Пример:

Да се превърне шестнадесетичното число **F31** в двоично.

Значи следваме алгоритъма: всяко едно от числата /буквите/ го превръщаме в тетради - **F** - > 1111 ; **3** $\rightarrow 0011$; **1** $\rightarrow 0001$. Сглобяваме получените тетради и получаваме двоичното число: $F31_{16} = 111100110001_2$.

Превръщане от двоична в шестнадесетична бройна система

Превръщането от двоична в шестнадесетична бройна система става обратно на шестнадесетично-двоичното преобразуване: числото в двоичен код се разделя на тетради (от дясно на ляво), ако отпред не стигат цифри се допълват нули, и съответните тетради се превръщат в шестнадесетични символи, пак с помощта на горната [таблица](#). Сглобяват се получените шестнадесетични символи и се получава съответното шестнадесетично число.

Пример:

Да се превърне числото **10110111101** от двоично в шестнадесетично.

Разделяме числото на тетради:

101 | 1011 | 1101

От пред се допълва една нула, за да се образува тетрада:

0101 | 1011 | 1101

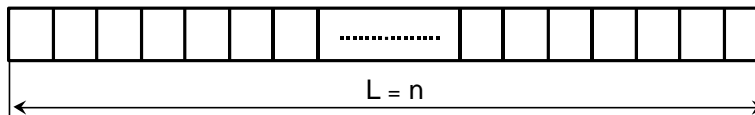
Съответните тетради се превръщат в шестнадесетични символи: **0101** → **5**; **1011** → **B**; **1101** → **D**. Сглобяват се шестнадесетичните символи и се получава шестнадесетичното число: **10110111101**₂=**5BD**₁₆.

Таблица на някои числа в различните бройни системи

БРОЙНА СИСТЕМА			
16 (HEX)	10 (DEC)	8 (OCT)	2 (BIN)
0	0	0	0000
1	1	1	0001
2	2	2	0010
3	3	3	0011
4	4	4	0100
5	5	5	0101
6	6	6	0110
7	7	7	0111
8	8	10	1000
9	9	11	1001
A	10	12	1010
B	11	13	1011
C	12	14	1100
D	13	15	1101
E	14	16	1110
F	15	17	1111

1.2 Понятие за разрядна мрежа

Мястото (полето), предназначено за записване (представяне) на числата, може да се определи (опише) като шаблон (рамка, дисплей). Това поле има дискретна структура, чиито елементи са отделните разряди, ето защо полето се нарича **разрядна мрежа** (PM). Основната характеристика на разрядната мрежа (фигура 1.1.6.1.1) е нейната дължина, която се измерва с броя на разрядите.

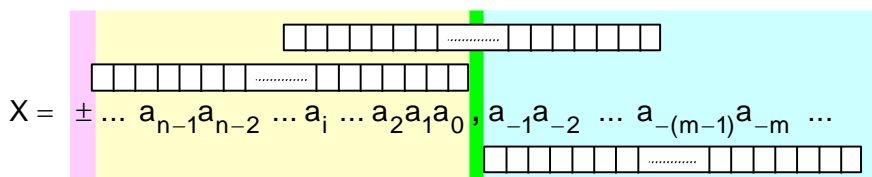


Фиг. 1.1.6.1.1. Представа за разрядна мрежа

Онази част от разрядната мрежа, в която може да се помести една цифра на числото, се определя като *един разряд*. В случаи, когато числото е двоично, разрядът побира една двоична цифра и неговият размер се определя на 1 [bit]. Така дължината на разрядната мрежа може да бъде определена в битове – $L=n[b]$. Тъй като съдържанието на разрядната мрежа се интерпретира според основата на бройната система, следва да се различават такива изказвания като: "*n разряда*" и "*n бита*". Определяйки дължината на разрядната мрежа в общия случай на n разряда, може да се твърди, че броят на представимите в нея числа, е равен на броя на възможните в нея цифрови комбинации, които се определят чрез уравнение (1.1.3.3).

Имайки предвид естествения вид на числото (1.1.6.1.1), възниква въпросът как да положим върху него шаблона на разрядната мрежа?

По принцип са възможни следните три положения:



Фиг. 1.1.6.1.2. Възможни положения на шаблона на PM

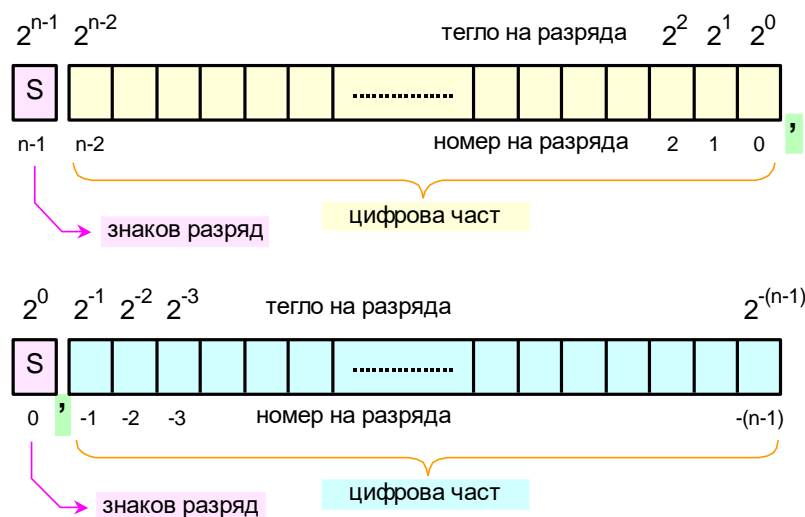
Първото положение (вижте горния правоъгълник) е такова, че то в най-висока степен съответства на естествения вид на числото, защото в разрядната мрежа ще се представят числа, имащи както цяла, така и дробна част.

Второто положение (вижте средния правоъгълник) се характеризира с това, че при него в разрядната мрежа ще се представят само цели числа. Възможностите му в сравнение с първото положение са много по-големи, но затова пък представяните числа няма да имат дробна част, т.е. ще бъдат по-неточно представяни.

Третото положение (вижте долния правоъгълник) е такова, че разрядната мрежа е разположена само върху дробната част, което означава, че в нея ще се представят числата по-малки от единица, при това много по-точно.

В крайна сметка практическо приложение намират последните две положения, при които запетаята се намира вън от разрядната мрежа. Имитацията на естествения вид на числото в разрядната мрежа, т.е. съчетаването и на двете части на числото, се оказва на практика неудобно, тъй като те са свързани в повечето случаи с различни алгоритми при опериране с тях. Пример за това са алгоритмите от [пункт 1.1.2](#).

И така, разрядната мрежа в изчислителните устройства най-често е така организирана, че положението на запетаята на числата спрямо нея е **фиксирано и неизменно** – отдясно или отляво. В този смисъл говорим за разрядни мрежи с дясно фиксирана запетая (ДФЗ) и за разрядни мрежи с ляво фиксирана запетая (ЛФЗ). Структурата на двата вида организация е илюстрирана на фигура 1.1.6.1.3.



Фиг. 1.1.6.1.3. Структура на двоична разрядна мрежа

Номерацията на разрядите, техните тегла, положението на запетаята, съдържанието на знаковия разряд – всичко това е една условна интерпретация на съдържанието на разрядната мрежа, т.е. на онова поле, представено на фигура 1.1.6.1.1. За число представено в разрядна мрежа с показаната по-горе структура говорим, че е във **форма с фиксирана запетая**.

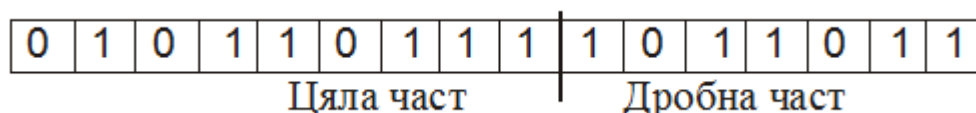
1.3 Числа с дясно и с ляво фиксирана запетая

Представяне на числата с фиксирана запетая

В числата отделянето на цялата част от дробната се извършва със синтактически знак - десетична точка ".". В литературата вместо десетична точка се използва термина десетична запетая, независимо, че се записва десетична точка.

Числата в оперативната памет могат да се въвеждат в полета с предварително определяне на мястото на разделителната запетая. В този случай е възприето да се посочва, че разделителната запетая е фиксирана.

При запис на числата с фиксирана точка в двоичен формат се предполага точно определено място на позиционната точка в полето за запис на числата. В битовете преди позиционната точка се записва цялата част на числото, а в битовете след нея – дробната (десетична) част (фиг.7.1). Както се вижда, мястото на позиционната точка не изисква специална клетка в паметта (бит) – тя е условно разделяне на полето на две части. Частен случай на този вид запис са целите числа. При тях позиционната точка е в края на полето. В практиката този вид на представяне на реалните числа се използва, като позиционната точка се фиксира в началото на полето за запис на числата.



Фиг.7.1 Представяне на реални числа с фиксирана точка

В този формат, числата се разполагат в полето спрямо разделителната точка и по-конкретно целите числа се разполагат от дясно на ляво, а дробната част от ляво на дясно.

Числата с фиксирана запетая се представят със стандартна дължина на полетата - 2,4 или 8 байта.

Фиксираната запетая не заема място в полето, а се "помни" обикновено от системата, т.е. системата се грижи за привеждане на резултатите съобразно предварително определеното място.

Една част от числата се представят с фиксирана запетая след най-младшият разряд на числата, т.е. в края на полето. В този случай се приема, че ще се представят цели числа.

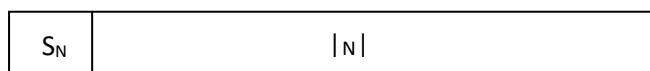
В поле от 4 байта могат да се поместват цели двоични числа с максимална разрядност от 31 бита. При опит да се въведе число с по-голяма разрядност се получава програмна грешка "препълване". Когато в поле за цели числа се въвеждат числа с дробна част, последната се отхвърля.

Има компютри от по-старо поколение, което поддържа полета с фиксирана запетая във старшият разряд, което означава, че числата, които се поместват в този тип полета ще се представят като числа по-малки от единица.

1.4 Прав, обратен и допълнителни кодове

Представяне на целите числа със знак.

Целите числа се представят обикновено в една машинна дума, /дължината и обикновено е кратна на 8 бита/, като се използва двоична бройна система. Цяло число N се представя по формата на фиг. 1, като най-левия разряд се използва за кодиране на знака, а останалите – за представяне на абсолютната стойност на числото.



Фиг. 9.1 Представяне на цели числа със знак в прав код

Прав, обратен и допълнителен код за представяне на числа със знак.

Микропроцесорът в компютърните системи изпълнява прости команди при аритметичната обработка на числови данни. За целта числовите данни се представят в няколко вида машинни кодове.

1. Прав код.

Правият код на двоично число се образува от абсолютната стойност на числото в двоична бройна система и код за знака (нула или единица). Той е неудобен за събиране на числа с различни знаци, затова в компютрите се използва т.нар. допълнителен код.

$X_{пк} = X$ (за положителни и отрицателни числа)

Пример:

$N(10) = 11 \rightarrow N(2) = 1011 \rightarrow N(2)_{пк} = 0 | 1011$

$N(10) = -15 \rightarrow N(2) = -1111 \rightarrow N(2)_{пк} = 1 | 1111$

2. Обратен код

Обратният код на двоичните числа се образува по следното правило: обратният код на положителните двоични числа съвпада с техният прав код; обратният код на отрицателните числа съдържа '1' в бита за знак, а цифрите на числото се заменят с техните инвертирани стойности (нулите се заменят с единици и обратно).

$$X_{ок} = \begin{cases} x & \text{при } 0 \leq x \leq 2^{n-1} - 1 \text{ (положителни числа)} \\ 2^n + x - 1 & \text{при } -2^{n-1} + 1 \leq x \leq 0 \text{ (отрицателни числа)} \end{cases}$$

където n е разрядността на машинната дума.

Пример:

$N(10) = 6 \rightarrow N(2) = 0110 \rightarrow N(2)_{ок} = 0 | 0110$

$N(10) = -13 \rightarrow N(2) = -1101 \rightarrow N(2)_{ок} = 1 | 0010$

Едно от свойствата на обратния код е че сумата на положително с обратния код на отрицателното му число дава така наречената машинна единица 111111111. Нулата в обратен код

има двойко значение: 0|0000 и 1|1111. Това е причина да не се използва обратен код в машинната аритметика, а той да се модифицира в допълнителен код.

3. Допълнителен код.

В допълнителен код положителните двоични числа съвпадат с техния прав код. Допълнителният код на отрицателните числа се образува, като към обратния код на числото се прибави '1' в най-младшия разред (бит).

Допълнителния код X_{dk} на едно число x се определя по формулата:

$$X_{dk} = \begin{cases} x & \text{при } 0 \leq x \leq 2^{n-1} - 1 \text{ (положителни числа)} \\ 2^n + x & \text{при } -2^{n-1} \leq x < 0 \text{ (отрицателни числа)} \end{cases}$$

където n е разрядността на машинната дума.

Използването на допълнителен код има това предимство, че алгебричната сума на 2 цели числа, независимо от техните знаци, се свежда до намирането на аритметичната сума на допълнителните им кодове, разглеждани като цели неотрицателни двоични числа.

Пример:

$N(10) = 19 \rightarrow N(2) = 10011 \rightarrow N(2)_{dk} = 0|10011$
 $N(10) = -13 \rightarrow N(2) = -1101 \rightarrow N(2)_{dk} = 1|0011$

Основни свойства на допълнителния код:

- Сумата на положително число с допълнителния код на неговото отрицателно значение дава така наречената машинна единица в допълнителен код МЕДК = 10|00000. Характерното в този запис е, че в битовете за знак има 10.
- Представянето на отрицателните числа в допълнителен код се явява допълнение до на правия код до машинната единица. Значението на допълнителния код може да се види от записа на последователните числа при прехода от положителни към отрицателни стойности:

Дес.число	5	4	3	2	1	0	-1	-2	-3	-4
Двоично число	0 0101	0 0100	0 0011	0 0010	0 0001	0 0000	1 0001	1 0010	1 0011	1 0100
Доп. код	0 0101	0 0100	0 0011	0 0010	0 0001	0 0000	1 1111	1 1110	1 1101	1 1100

От таблицата може да се види, че ако към някое число, представено в допълнителен код, прибавим 1 към най-младшия бит се получава предхождащото го число, независимо дали е положително или отрицателно. Това свойство на допълнителния код на двоичните числа улеснява извършването на аритметичните действия с тях.

Да обърнем внимание, че правият, обратният и допълнителният кодове на неотрицателните числа са едни и същи.

Задачи за самостоятелна работа:

Намерете допълнителния код на числата:

- 32; -32
 45; -45
 96; -96
 136; -136

1.5 Събиране и изваждане на числа с дясно фиксирана запетая

Събиране и изваждане на числа **без знак**.

Събиране на двоични числа

Събирането на две цели положителни числа, записани в двоична система, извършваме по аналогия със събирането в десетична система – чрез събиране на едноименните единици, означени със съответните цифри, с евентуално получаване на “едно наум”, т.е. на една единица от по-висок ред, която пренасяме към съответните единици.

При събиране на двоични числа се извършват следните действия:

1. Числата се записват едно под друго.
2. Събирането се извършва отдясно наляво.
3. Събират се цифрите в съответните позиции, като се започва от нулева позиция.

Ако полученото число е 0 или 1, то се записва в съответната позиция на сумата.

В противен случай като резултат се записва 0 и се прави пренос от една 1 в съседната лява позиция.

Като резултат от събирането на:

- ☐ две единици се получава **0** и пренос в предната (лява) позиция **1**;
- ☐ три единици се получава **1** и пренос в предната (лява) позиция също **1**;
- ☐ четири единици се получава **0** и пренос в предната (лява) позиция **две единици**;
- ☐ пет единици се получава **1** и пренос в предната (лява) позиция **две единици** и т.н.

4. Тези стъпки се извършват последователно за всички събираеми цифри и преноси за всяка следваща позиция на числата.

Примери:

Задача1. Съберете числата **10010₍₂₎** и **1011₍₂₎**.

Задача2. Съберете числата **1011₍₂₎** и **11001₍₂₎**.

$$\begin{array}{r} \textcolor{red}{1} \\ 1\ 0\ 0\ 1\ 0 \\ + \\ 1\ 0\ 1\ 1 \\ \hline 1\ 1\ 1\ 0\ 1 \end{array}$$

$$10010_{(2)} + 1011_{(2)} = ?_{(2)} = \mathbf{11101_{(2)}}$$

$$\begin{array}{r} \textcolor{red}{1\ 1\ 1} \\ 1\ 0\ 1\ 1 \\ + \\ 1\ 1\ 0\ 0\ 1 \\ \hline 1\ 0\ 0\ 1\ 0\ 0 \end{array}$$

$$1011_{(2)} + 11001_{(2)} = ?_{(2)} = \mathbf{100100_{(2)}}$$

Правила за събиране на две цифри в двоична бройна система можем да запишем във вид на следната таблица за събиране в двоична система:

+	0	1
0	0	1
1	1	10_2

Тази таблица ще прилагаме за намиране двоичния запис на сумата на две числа, записани в двоична система.

Започваме с най-дясната позиция, т.е. с цифрите от първи разред. Ако в двете събираеми имаме цифри 0 и 0, в резултата пишем цифрата 0. Ако цифрите са съответно 0 и 1 или 1 и 0, ще запишем цифрата 1. Ако имаме цифри 1 и 1, в същата позиция ще запишем цифрата 0 и тъй като получаваме “едно наум”, правим пренос, като в съседната вляво позиция прибавяме една единица и продължаваме определянето на следващите цифри само с тези операции.

За пояснение ще разгледаме следния пример за събиране:

$$\begin{array}{r}
 1101011_2 \\
 + 110010_2 \\
 \hline
 10011101_2
 \end{array}$$

Тук най-десните цифри (цифрите от първи разред – цифрите в първа позиция на двете събираеми) са съответно 1 и 0 и на тях съгласно таблицата за събиране отговаря цифрата 1. Следващите две цифри са 1 и 1 – пишем 0 и правим пренос наляво с една позиция, имаме “едно наум”. В третата позиция имаме цифрите 0 и 0, на които отговаря цифрата 0, но тук е пренесена една единица, следователно имаме съпоставяне на цифрата 0 с цифрата 1 - на тези цифри ще отговаря съгласно таблицата цифрата 1, която пишем в резултата като цифра от трети разред. В четвърта позиция имаме цифрите 1 и 0 – пишем 1, в петата позиция на цифрите 0 и 1 също отговаря 1. В шестата позиция се намират цифрите 1 и 1. На тях отговаря цифрата 0 и правим пренос на една единица наляво. Там първото събираемо има за последна цифра цифрата 1, а второто събираемо няма цифра, но за пълнота можем да си мислим (и пишем) цифрата 0 и на тях ще съответства цифрата 1, а на нея и пренесената единица ще отговаря 0 и ще имаме последен пренос, който направо отразяваме в резултата с най-лява цифра 1.

$$\text{И така: } 1101011_2 + 110010_2 = 10011101_2$$

Проверка в десетична система:

$$1101011_2 = 64+32+8+2+1=107_{10}$$

$$110010_2 = 32+16+2=50_{10}$$

$$10011101_2 = 128+16+8+4+1=157_{10}$$

$$\text{Очевидно } 107_{10}+50_{10}=157_{10}$$

Събирането на повече от две числа се основава на последователното събиране на числата две по две и може да се извършва едновременно, както в следния

пример:

$$\begin{array}{r} 11101_2 \\ + 10111_2 \\ 11111_2 \\ \hline 1010011_2 \end{array}$$

т.е. $11101_2 + 10111_2 + 11111_2 = 1010011_2$

Задачи за самостоятелна работа:

Съберете следните двоични числа:

$$100101_2 + 10110_2$$

$$1011101_2 + 110101_2$$

$$111_2 + 1011_2 + 1101_2$$

$$101111_2 + 101010_2$$

$$1101_2 + 111101_2$$

$$101_2 + 1111_2 + 11101_2$$

Изваждане на двоични числа.

Изваждането на двоични числа е аналогично с това на десетични числа.

Примери:

Задача1. Извадете числата $101101_{(2)}$ и $10100_{(2)}$.

$$101101_{(2)} - 10100_{(2)} = ?_{(2)} = 11001_{(2)}$$

$$\begin{array}{r} \overset{\cdot}{1}01101 \\ - 10100 \\ \hline 11001 \end{array}$$

Задача2. Извадете числата $1111101_{(2)}$ и $101111_{(2)}$.

$$1111101_{(2)} - 101111_{(2)} = ?_{(2)} = 1001110_{(2)}$$

$$\begin{array}{r} \overset{\cdot}{1}\overset{\cdot}{1}\overset{\cdot}{1}1101 \\ - 101111 \\ \hline 1001110 \end{array}$$

*Действието **изваждане** е обратно на събирането –разликата на умаляемото и умалителя, събрана с умалителя, трябва да дава умаляемото.*

Оттук следва, че ще прилагаме същата таблица за събиране и ще се налага да правим пренос, но в обратен ред – това ще бъде предварителен пренос отляво надясно и ще означава, че една единица от даден ред се разлага на две единици от съседния по-нисък ред. Следващият пример илюстрира изваждането на две числа:

$$\begin{array}{r}
 100101_2 \\
 - \quad 1110_2 \\
 \hline
 010111_2 \\
 \dots\dots\dots
 \end{array}$$

Тук с точки над цифрите сме означили промяната в единиците поради пренос надясно. Тъй като във втората позиция на двоичния запис на умаляемата цифрата е 0 – няма единица от първи ред, пренасяме наличната единица от втори ред, т.е. числото 22, от трета позиция във втора, където тя се разглежда като две единици от първи ред – $2^2 = 2^1 + 2^1$, и тъй като изваждаме едната от тях, остава само една и в резултата на втора позиция пишем цифрата 1. С подобни разсъждения намираме и останалите цифри:

$$110101_2 - 1110_2 = 10111_2$$

$$\text{тъй като } 110101_2 = 1110_2 + 10111_2$$

Задачи за самостоятелна работа:

Извършете изваждането:

$$100_2 - 11_2$$

$$100001_2 - 10110_2$$

$$101101_2 - 10100_2$$

$$1111101_2 - 101111_2$$

$$101001_2 - 110111_2$$

$$110011_2 - 11111_2$$

$$1000111_2 - 111011_2$$

$$10100101_2 - 111111_2$$

$$10101101_2 - 1010111_2$$

$$10000000_2 - 1111111_2$$

Събиране и изваждане на числа **СЪС ЗНАК**.

Събиране и изваждане на числа в допълнителен код.

Основното преимущество на двоичния допълнителен код е, че събирането и изваждането на всички числа (положителни и отрицателни, 0) се извършва по един и същи алгоритъм.

Аритметичните действия се правят както в обикновената двоична аритметика. За целта числата трябва да са с еднаква дължина, като резултатът е със същата дължина. В този случай всеки бит на пренос в най-лявата част (където е бита за знак) се игнорира. Когато числото е с по-малък размер то се допълва отляво с незначещи нули. Действието изваждане се представя като събиране, но второто число се представя като отрицателно число в допълнителен код

Пример за събиране и изваждане на числа със знак.

Зад.1 Намерете $26-59=?$

Ще извършим действие изваждане между положителни числа представени в допълнителен код:

$$59_{(10)}=00111011_{(2)}\text{ПК} = 00111011=\text{ОК}=\text{ДК}$$

$$26_{(10)}=00011010_{(2)}\text{ПК} = 00011010=\text{ОК}=\text{ДК}$$

Изваждаме ги двоично в 8 бита по правилата на двоичната аритметика, отчитайки възникналите преноси/заеми. За проверка ще извършим изваждане и между десетичните числа:

26 =	0 0 0 1 1 0 1 0 ДК=ПК
-	-
59 =	0 0 1 1 1 0 1 1 ДК=ПК
-----	-----
-33	1 1 0 1 1 1 1 1 ДК

Резултатът се отрязва само до 8 бита, заемът от b8 не се взема предвид /ако няма препълване действията в ДК дават резултат с дължина колкото операндите/8б/, който също е представен в ДК.

Проверка:

11011111 ДК
-
1

11011110 ОК
10100001 ПК

Преобразуване в ПК на отрицателния резултат, за да се види кое число представя

Извадохме 1 от ДК
Инвертирахме битовете на ОК без знаковия бит

Двоичната разлика $10100001_{\text{ПК}} = -(2^5 + 2^0) = -33$ и представя получената десетична сума при събирането на числата в десетичен код.

Може да смените действие изваждане със събиране, но пак в ДК и пак се получава същия резултат: $26 - 59 = 26 + (-59)$. Този път трябва да представите -59 в ДК и да събирате, вместо да изваждате.

Зад.2 Да се преобразуват в ДК числата -105 и 18 и представят в 8 бита и в 16 бита, после да се сумират и да се извадят. Направете проверка.

105	1	$105 = 1101001_{(2)}$
52	0	
26	0	
13	1	
6	0	
3	1	
1	1	
0		

18	0	$18 = 10010_{(2)}$
9	1	
4	0	
2	0	
1	1	
0		

<u>-105</u>	8b	16b	
ПК	11101001	10000000	01101001
ОК	10010110	11111111	10010110
	+ 1	+	1
ДК	10010111	11111111	10010111

<u>18</u>	8b	16b	
ПК	00100010	00000000	00100010
ОК	00100010	00000000	00100010
ДК	00100010	00000000	00100010

-105 _{ДК}	=10010111	11111111	10010111
+			
18 _{ДК}	=00010010	00000000	00010010

-87	10101001	11111111	10101001 _{ДК}

Проверка:

11111111 10101001 _{ДК}	Преобразуване в ПК на отрицателната сума:
11111111 10101000 _{ОК}	Извадихме 1 от ДК
10000000 01010111 _{ПК}	Инвертирахме битовете на ОК без знаковия
Сумата е 10000000 01010111 _{ПК} $= -(2^6 + 2^4 + 2^2 + 2^1 + 2^0) = -87$	

-105 _{дк} =10010111	11111111	10010111
-		
18 _{дк} =00010010	00000000	00010010

-123 10000101	11111111	10000101 _{дк}

Проверка:

11111111 10000101 _{дк}	Преобразуване в ПК на отрицателната разлика:
11111111 10000100 _{ок}	Извадихме 1 от ДК
10000000 01111011 _{пк}	Инвертирахме битовете на ОК без знаковия

Разликата е 10000000 01111011_{пк} = $-(2^6+2^5+2^4+2^3+2^2+2^0)=-123$

Зад. 3. Преобразувайте в ПК, ОК, ДК в 8 и 16 бита числата +117 и -117. Сравнете получените допълнителни кодове.

Зад.4. Да се преобразуват в ДК числата -135 и 129 в 9 и в 16 бита. Да се сумират и извадят. Направете проверка.

135	1	135=10000111 ₍₂₎
67	1	
33	1	
16	0	
8	0	
4	0	
2	0	
1	1	
0		

129	1	129=10000001 ₍₂₎
64	0	
32	0	
16	0	
8	0	
4	0	
2	0	
1	1	
0		

-135			
ПК	110000111	10000000	10000111
ОК	101111000	11111111	01111000
+	1	+	1
ДК	101111001	11111111	01111001

129			
ПК	010000001	00000000	10000001
ОК	010000001	00000000	10000001
ДК	010000001	00000000	10000001

Извършете самостоятелно означените действия!!!

1.6 Дефиниция за препълване и начини за неговото откриване.

Операциите събиране и изваждане на числа със знак при формат с фиксирана запетая се изпълняват с използване на допълнителен код, който се използва като машинен код на числата. При тези операции е възможно да се получи грешен резултат, който се дължи на преноси в най-старшия цифров разряд и в знаковия бит. В тези ситуации говорим за настъпило препълване, което се отбелязва чрез признака за препълване **V**. Препълването се изявява по две паралелни и различни линии. В единия случай се получава абсурдното съчетание на знаците на операндите и знака на резултата, а в другия случай препълването се обосновава чрез възможните преноси в и от знаковия разряд. Ето защо разпознаване на препълването се постига с помощта на логическа функция, синтезирана върху тази логика.

А) при числа **без знак** препълване има при възникване на пренос от най-старшия разряд, т. е.

$$V_0 = p_n . \quad (92)$$

Б) при числа **със знак** препълване се открива по три различни начина:

Б1) чрез знаците на числата ($Z=X+Y$) .

$$V_1 = (x_{n-1} \cap y_{n-1} \cap \overline{z_{n-1}}) \cup (\overline{x_{n-1}} \cap \overline{y_{n-1}} \cap z_{n-1}) . \quad (93)$$

Б2) чрез преносите в и от знаковия разряд на кодовата сума:

$$V_2 = p_{n-1} \oplus p_{n-2} . \quad (94)$$

Б3) чрез знаковите разряди на модифицирания код на кодовата сума:

$$V_3 = z_n \oplus z_{n-1} . \quad (95)$$

Различава се препълване **отляво (отрицателно)** и препълване **отдясно (положително)**.

ПРИМЕР. При събиране в допълнителен код в разрядна мрежа с дължина $n=5[b]$ на избраните по-долу числа, ще се наблюдава:

Препълване отляво

$$Z = (-12) + (-13)$$

Препълване отдясно

$$Z = 12 + 13$$

	1 0	← преноси
$[-12]_{\text{ДК}} =$	1 0100	
	+	
$[-13]_{\text{ДК}} =$	1 0011	
	0 0111	
	0 1 1	← преноси
$[12]_{\text{ДК}} =$	0 1100	
	+	
$[13]_{\text{ДК}} =$	0 1101	
	1 1001	

За така изпълнените операции препълването може да се разпознае по стойностите на логическите функции (93) и (94):

$$V_1 = (1.1.1) \cup (0.0.0) = 1;$$

$$V_1 = (0.0.0) \cup (1.1.1) = 1;$$

$$V_2 = 1 \oplus 0 = 1;$$

$$V_2 = 0 \oplus 1 = 1.$$

При препълване, получената сума се счита **за неверен резултат !**