ХИМИКОТЕХНОЛОГИЧЕН И МЕТАЛУРГИЧЕН УНИВЕРСИТЕТ – СОФИЯ

ИНФОРМАТИКА

част първа

лектор: гл. ас. д-р Стефан М. Панов

Катедра “Информатика”

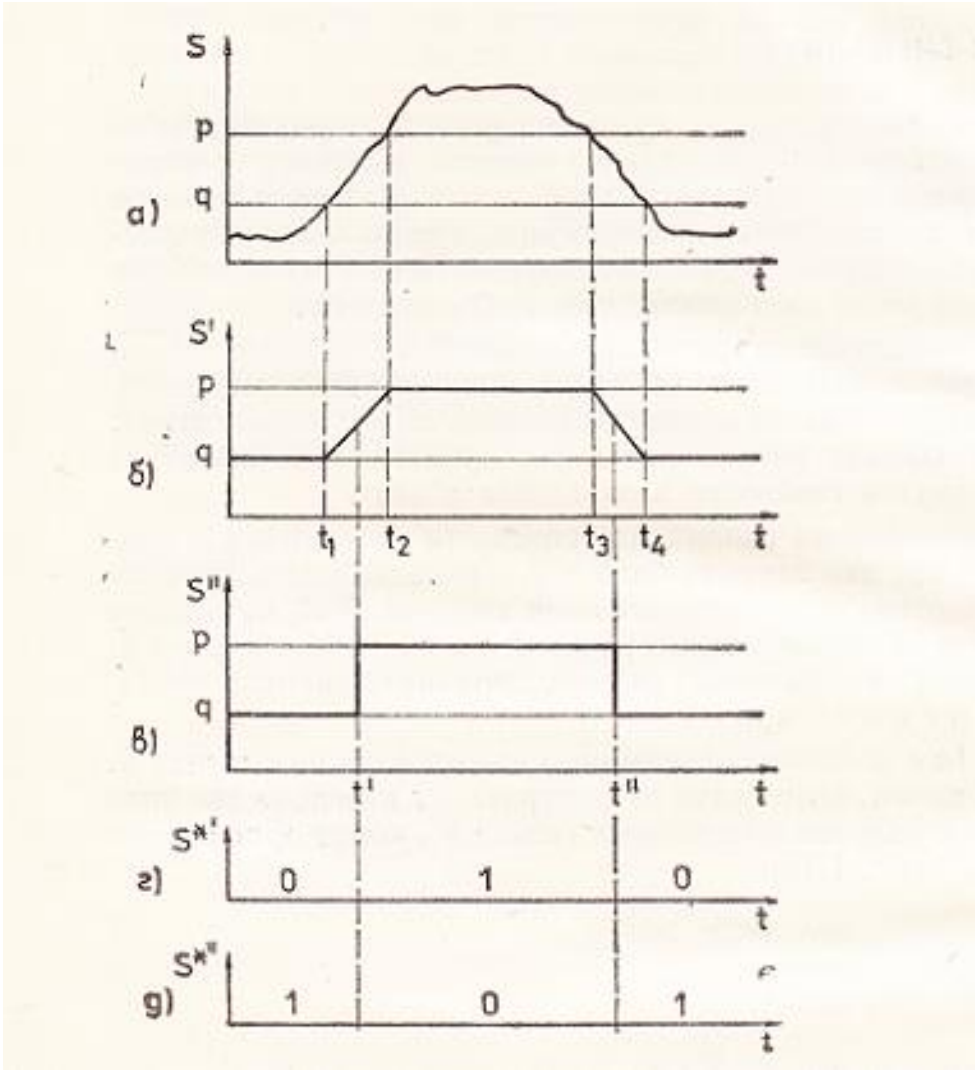
Лекция 1

АРИТМЕТИЧНИ И ЛОГИЧЕСКИ ОСНОВИ НА КОМПЮТЪРНИТЕ СИСТЕМИ

1.1 Кодиране на информацията в компютърните системи. Двоични сигнали.

Разграничаването на различните стойности на сигнала е най-сигурно и просто, ако те са две на брой. Сигнали, които приемат само две стойности, се наричат **двоични сигнали**. Тъй като двете стойности могат да съответстват само на двете логически стойности „**истина**” и „**лъжа**”, **двоичните сигнали се наричат логически**.

Идеализираният сигнал е неопределен в интервалите $t_1 < t < t_2$ и $t_3 < t < t_4$. В тия интервали от време сигналът променя своята стойност. При по-нататъшната идеализация, фиг. 1.1в, допускаме, че преминаването на сигнала между двете нива се осъществява мигновено ($\tau = 0$), което води до абстрахиране от интервалите на неопределеност и се получава **идеален двоичен сигнал**.



а) реален двоичен сигнал

б) идеализиран двоичен сигнал

q, ако S<=q

S‘ = p, ако S>=p

неопределен, ако q<S<p

в) идеален двоичен сигнал

г) право кодиране

S*‘ = 0, ако S*‘=q

S*‘ = 1, ако S*’=p

д) обратно кодиране

S*‘‘ = 1, ако S*’’=q

S*‘‘ = 0, ако S*’’=p

Фиг. 1.1. Идеализация на реален двоичен сигнал

1.2 Бройни системи (БС)

Бройната система е начин за представяне на числата с помощта на набор от символи, имащи определени **количествени** значения.

Тия символи се наричат ***цифри***. За представяне на числата **различните** бройни системи използват **различен** набор от цифри. Освен цифрите, БС включват и ***правила*** за представяне на числата с цифри.

Видове бройни системи:

- позиционни
- непозиционни

В позиционните БС **всяка цифра има определено тегло**, зависещо от **позицията** на цифрата в числото.

Броят на цифрите, които съдържа позиционната БС, се нарича ***основа*** на бройната система.

В непозиционните БС броят и позициите на цифрите в числото **не определят** неговата големина.

Примери за позиционни БС

Десетична БС (арабска)

Цифри: 0,1,2,3,4,5,6,7,8,9, Основа: 10

Пример 1: $\mathbf{5187 = 5.10^3 + 1.10^2 + 8.10^1 + 7.10^0}$

Пример 2: $\mathbf{3.46 = 3.10^0 + 4.10^{-1} + 6.10^{-2}}$ **Извод:**

Вляво от десетичната запетая се намира нулевата степен

Обобщение: Вляво от запетаята се намира нулевата степен.

Степените нарастват наляво, намаляват надясно.

Примери за непозиционни БС

Римска БС

Цифри:	*I ,*	*V,*	*X,*	*L,*	*C,*	*D,*	*M*
	1	*5*	*10*	*50*	*100*	*500*	*1000*

Вместо цифри се използват латински букви.

Пример 1: **CVIII = 100 + 5 + 1 + 1 + 1 = 108**

Пример 2: **IX = 10 - 1 = 9**

Двоична бройна система

Цифри: 0,1 **Основа: 2**

Двоичната система е **най-простата** бройна система. Тя е и най-лесна за използване в изчислителните системи. Устройствата, които представят числата, трябва да имат две устойчиви състояния, отговарящи на двете цифри от системата 0 и 1. Това свойство на двоичната бройна система е причина в съвременните изчислителни системи да се използва само двоична информация.

Пример 1: $101101_2 = 1.2^5+0.2^4+1.2^3+1.2^2 + 0.2^1+1.2^0$

$= 32 +0 + 8 +4 +0 +1 = 45$

Пример 2: $1,101_2 = 1.2^0 + 1.2^{-1} + 0.2^{-2} + 1.2^{-3}$

$= 1 + 1/2+ 0 +1/8 = 0.5 + 0.125 = 1,625$

Преобразуване на двоично число в 10-ично:

Ако събираемите в горните представяния се изчислят и сумират, ще се получи 10-ичният еквивалент на двоичното.

Забележка: За да е ясно в каква БС се записва числото, основата (ако не е 10) се записва като индекс към числото: Например 1101001_2 е двоично число.

Шестнадесетична бройна система (*основа 16*)

Цифри: 0,1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

Буквените комбинации от A до F заместват двуцифрените десетични числа от 10 до 15.

Пример 1: $3FD_{16} = 3.16^2+15.16^1+13.16^0$

$= 768 + 240 + 13 = 1021$

Пример 2: $C,8A_{16} = 12.16^0 + 8.16^{-1} + 10.16^{-2}$

$= 12 + 8/16 + 10/256$

$= 12,5390625$

Преобразуване на цели десетични числа в бройна система с основа b >= 2

Алгоритъм 1 (за цели числа) :

1. Числото се дели на основата **b** до получаване на частно и остатък. Остатъкът се запомня.
2. Ако полученото частно е по-малко от **b**, то също е остатък и се отива към точка 4, иначе към точка 3.
3. Полученото частно отново се дели на **b** и се преминава към точка 2.
4. Числото в новата бройна система се записва като последователност от получените остатъци в ред, обратен на тяхното получаване.

Преобразуване на десетични дробни числа, по-малки от единица в БС с основа b >= 2

Алгоритъм 2:

1. Правилната дроб се умножава по основата **b** и цялата част, получена при умножението, се запомня.
2. Проверява се дали броят на извършените до момента умножения е равен на броя на цифрите в новата дроб, който се задава по условие. Ако е равен, се отива към точка 4, иначе към точка 3.

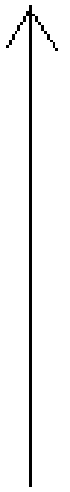
3. Получената дробна част се умножава отново по основата **b** и цялата част, получена при умножението, се запомня. Отива се към точка 2.
4. Правилната дроб в новата бройна система се записва като последователност от получените цели части по реда на тяхното получаване.

! От алгоритъма следва, че новото представяне почти винаги е приближение на началното число.

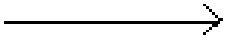
! За реалните числа, имащи цяла и дробна част се използват поотделно двата алгоритъма.

Пример: Преобразуване на числото 79 от десетична в двоична БС.

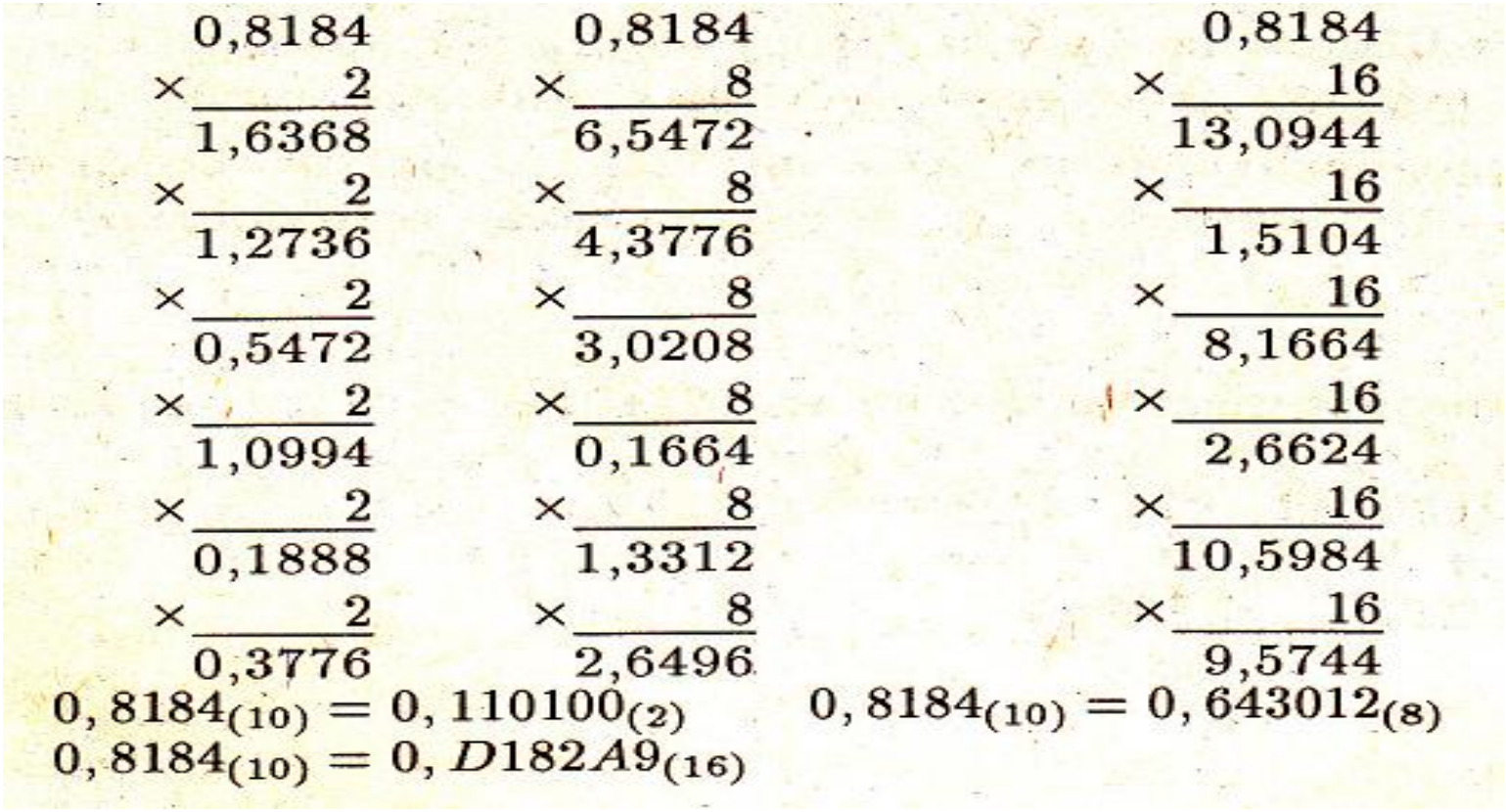
	Число в десе-тична БС	Основа на новата БС	Остатък от деленето
число	79	:2	1
частно	39	:2	1
частно	19	:2	1
частно	9	:2	1
частно	4	:2	0
частно	2	:2	0
частно	1	:2	1



$1001111_{(2)} = 79_{(10)}$



Пример: Да се преобразува дробта 0,8184 в двоично, осмично, шестнадесетично число.



Осмична бройна система

Цифри: 0,1, 2, 3, 4, 5, 6, 7

Основа: 8

Пример 1: $256_8 = 2.8^2+5.8^1+6.8^0$

$= 128 + 40 + 6 = 174$

Пример 2: $7,42_8 = 7.8^0 + 4.8^{-1} + 2.8^{-2}$

$= 7 + 4/8 + 2/64 = 7,53125$

Преобразуване 2 ⇔ 16 и 2 ⇔ 8

Шестнадесетично число	Осмично число	Двоично число	Десетично число
0	0	0000	0
1	1	0001	1
2	2	0010	2
3	3	0011	3
4	4	0100	4
5	5	0101	5
6	6	0110	6
7	7	0111	7
8	10	1000	8
9	11	1001	9
A	12	1010	10
B	13	1011	11
C	14	1100	12
D	15	1101	13
E	16	1110	14
F	17	1111	15

Правило: При преобразуване на двоично число в осмично всяка тройка двоични цифри се заменя със съответстващата ѝ осмична цифра. При обратното преобразуване всяка осмична цифра трябва да се замести със съответна тройка двоични цифри.

Пример 1: $10\ 011\ 111{,}1_2 = 010\ 011\ 111{,}100_2$

$2\quad 3\quad 7\ ,\quad 4 = 237{,}4_8$

Пример 2: $5\ 0\ 6{,}\ 7_8$

$101\ 000\ 110{,}\ 111_2$

Заб. 1 Групирането по тройки става спрямо двоичната запетая.

Заб. 2 При необходимост се добавят незначещи нули.

Правило: При преобразуване на двоично число в шестнадесетично всяка четворка двоични цифри се заменя със съответстващата ѝ шестнадесетична цифра. При обратното преобразуване всяка шестнадесетична цифра трябва да се замести със съответна четворка двоични цифри.

Пример 3: $1010\ 1011, 011_2 = 1010\ 1011, 0110_2$

$A \quad B \quad 6 = AB6_{16}$

Пример 4: $F\ 0\ 6, 7_{16}$

$1111\ 0000\ 0110, 0111_2$

Заб. 1 Групирането по 4-ки става спрямо двоичната запетая.

Заб. 2 При необходимост се добавят незначещи нули.

Двоична аритметика

Събиране:

0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 10 пренос

Изваждане

0 - 0 = 0
1 - 0 = 1
1 - 1 = 0
0 - 1 = 1 със заем

```
100101    37
 10110    22
----------------
111011    59
```

```
101101   45
 10100   20
---------------
 11001   25
```

1.3 Представяне на информацията в комп. системи

Понятие за бит – *определение:*

Най-малкото количество информация може да се разглежда като отговор на въпрос, който има два възможни варианта– да (true, 1) и не (false, 0), където 1 и 0 са едноцифрени двоични числа).

Информацията, съдържаща се в отговора на такъв въпрос се нарича '***бит***' (***bit***). (bit – binary digit – двоична цифра).

Алтернативно определение: Един бит служи за кодиране на едно от две възможни състояния или явления.

Двоичната информация се групира в множество от битове, наречени ***байтове (B – byte)***.

Един ***байт (B – byte)*** се състои от 8 бита.

Производни единици на байта са килобайт (1 KB = 1024 B), мегабайт (1 MB = 1024 KB), гигабайт (1 GB = 1024 MB), терабайт (1 TB = 1024 GB) петабайт (1 PB = 1024 TB) и т.н.

Производните единици KB, MB, GB, TB, PB се различават от стандартните измервателни единици – вместо ***множител*** 1000 се използва $\mathbf{2^{10} = 1024}$ (множителят е кратен на основата на двоичната БС). По тоя начин **най-икономично** се използва паметта в КС.

Представяне на числова информацията в КС

В компютърните системи се използват основно **два вида числа**: **цели** (натурални) числа и **реални** (веществени) числа.

А. Цели числа без знак – целите неотрицателни числа, които се представят в двоична форма чрез преобразуване по начина, показан по-горе в раздела БС.

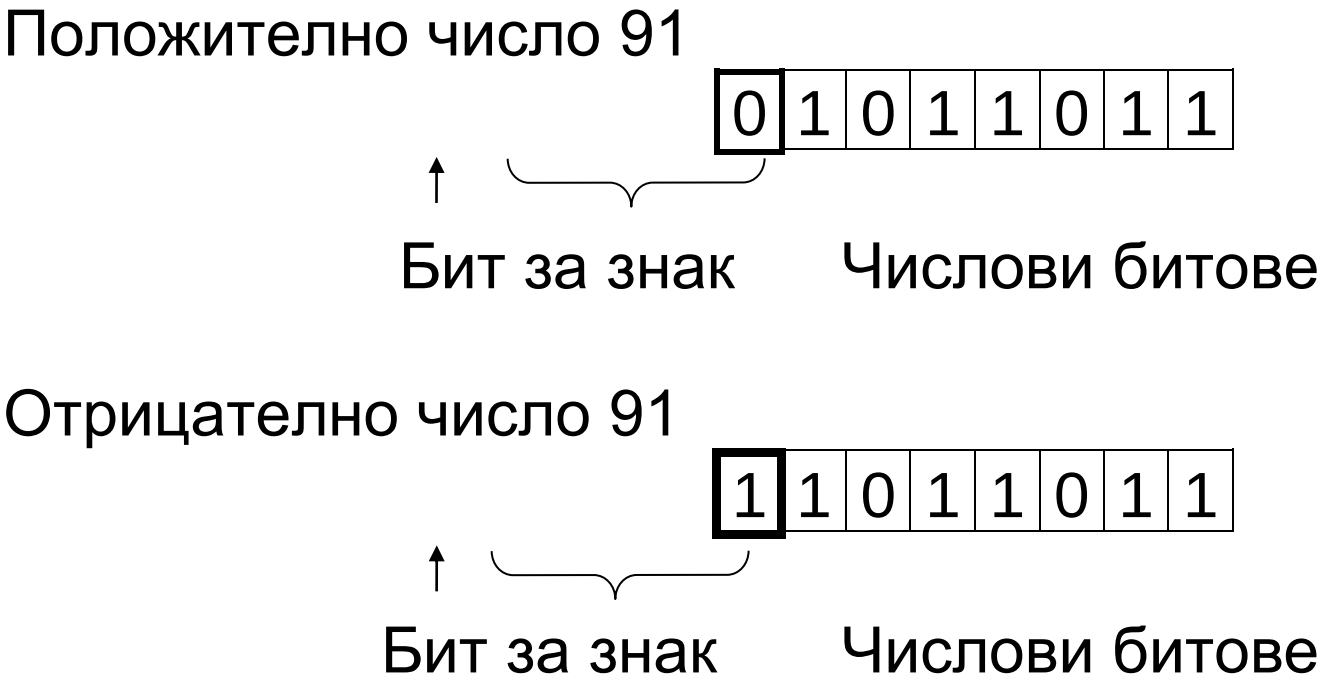
В зависимост от големината на числото за представянето му са необходими различен брой цифри (битове). В компютърните системи числата се представят с **фиксиран брой битове**. Т.е. числата се представят с определена **точност**.

Броят на целите числа, които могат да се съставят с помощта на **n двоични цифри,** е $\mathbf{2^n}$. Тогава **целите неотрицателни числа**, описани с **n** бита, са в интервала $\mathbf{0 \div 2^n - 1}$. В ***един байт*** (8 бита) може да се представи цяло неотрицателно число в интервала $\mathbf{0 \div 2^8 - 1}$ или **0÷255**.

Б. Цели числа със знак:

Представянето на тези числа изисква въвеждане на ***специален бит за знака*** на числото. Приема се, че ако в бита за знак има стойност 0, числото е положително, а ако стойността е 1, то е отрицателно. Така например, ако за цяло число се отделя 1 байт, стойността на числото се записва в последните 7 бита, а първият бит се използва за знак.

На фиг. 1.2 е показано разположението на цифрите на цяло число като положителна и отрицателна стойност.



Фиг. 1.2. Представяне на цели числа

Допълнителен код

Отрицателните числа се представят в допълнителен код. Той се получава от правия код на числото чрез инвертиране (обратен код) и добавяне на единица.

Правият код на числото е самото число представено в двоична бройна система, като най-старшият му бит (тоя най-вляво) е знаковия бит.

Пример: - 5_{10} = 00000101_2 прав код (представяне в 8 бита!)

1) Инвертираме =11111010 до обратен код
   +1
2) Прибавяме 1 = 11111011 допълнителен код (ДК)

Второ, по-лесно правило за преобразуване на едно положително число в отрицателно в допълнителен код:

Намира се най-дясната единица. Вляво от нея се сменят цифрите.

Същото правило се прилага и за преобразуване на едно отрицателно число (в допълнителен код) в положително.

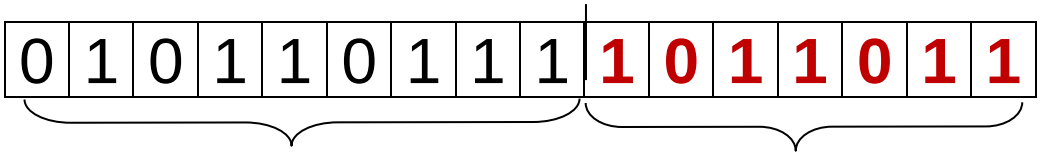
Т.е. за да установим на колко е равно едно отрицателно двоично число, записано в допълнителен код, прилагаме правилото, а после преобразуваме положителното число в десетично.

Чрез ДК операциите по изваждане (а=в-с) в КС се свеждат до събиране на числа в прав и ДК (а=в+(-с), където (-с) е в ДК.

Примери на упражнение.

Реални (вещественi) числа

В компютърните системи реалните числа се представят по два начина – с фиксирана и с плаваща запетая (точка). При запис на числата с фиксирана точка в двоичен формат се предполага точно определено място на позиционната точка в полето за запис на числата.
Цифрите вляво от позиционната точка представят цялата част на числото, а цифрите вдясно – дробната част.



Фиг. 1.3 Представяне на реални числа с фиксирана точка.

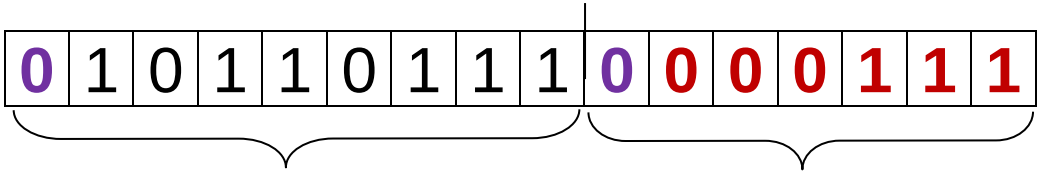
За числата с плаваща запетая се дефинират т. нар. мантиса **m** и порядък **p**.

Пример 1: в десетична БС:

-374.25 може да се запише във вида: -0.37425×10^3
където **m** = -0.37425, а **p** = 3.

Пример 2: в двоична БС: $1011011.1_2 = 0.10110111 \times 2^7$

Числата с плаваща точка се представят в паметта на компютърните системи, като в полето за запис на числото се фиксира част от паметта за запис на мантисата и друга част – за запис на порядъка (фиг. 1.4).



Фиг. 1.4 Представяне с плаваща точка за пример 2.

Мантисата и порядъкът се записват в паметта както се записват целите числа. Те трябва да имат по един бит за знак, а останалите битове от полето са за запис на стойността. Но ако искаме да подсигурим мантисата да бъде по абсолютна стойност по-малка от 1, можем да мислим, че двоичната точка се намира между най-старшия (знаковия) бит и неговия съсед.

Машинна нула и препълване – на упражнения

Представяне на символна информация в КС

В компютърните системи информацията се съхранява само в двоичен вид. Използваните символи също се представят като комбинация от двоични числа.

Така текстът, записан в паметта на компютъра, представлява последователност от ***байтове***, съответстващи на символите от текста.

Най-често в КС се използват ASCII и Unicode таблиците. **Дават съответствието между даден символ и неговото представяне като число в паметта на компютрите.** Те са се превърнали в стандарт, чрез който може да се обменя информация между различни КС.

ASCII (American Standard Code for Information Interchange)
Когато е разработван тоя стандарт, се е смятало, че всички използвани в изчислителните машини символи ***не надвишават 120*** на брой и могат да се кодират със седем двоични цифри: $2^7 = 128$. С цел подсигуряване е прието символите да записват с осем двоични цифри, което е позволявало да се кодират до 256 символа.
Пример: символ 0 1 2 A B C /
десетично число 48 49 50 65 66 67 47

Символите с номера над **128** се използват за ***други символи и азбуки***, например за символите на кирилицата.
Големият проблем е, че текстовове, писани на кирилица, ще изглеждат неразбираемо, когато се смени текущата кодировка.

Unicode

Поради нарастващата нужда от повече символи се въвежда системата за кодиране Unicode. С нея работят ***съвременните операционни системи***.

При тази система за представянето на един символ се използва не един байт (8 бита), а ***2 байта*** (16 бита). Това означава, че в Unicode могат да се кодират до $\mathbf{2^{16}= 65536}$ различни символа.
Първият байт на всеки символ от кирилицата в тая система съдържа като код на азбуката числото 204, а вторият байт е кода на символа/буквата.

Над 20000 символа в стандартните шрифтове в Unicode системата са отделени за йероглифи от китайското и японско писмо.

1.5 Логически основи. Базови понятия.

Теоретическите основи за създаването на ЕИМ (КС) са изградени върху фундамента на няколко специални математически дисциплини. Една от тях е логическата алгебра (**булева алгебра** - БА), разработена от англ. математик Джордж Бул. Апаратът на БА се използва за синтезиране и оптимизация на електронните схеми, използвани в комп. системи.

Съждение - всяко твърдение, за което може да се съди дали е истинно или не. **Примери**: **7 > 4; X > 5; „В момента вали“;**
Ако се абстрахираме от конкретното съдържание на съжденията и разглеждаме само какъв е отговора на въпроса “истина или лъжа?” те могат да се разглеждат като **абстрактни обекти (АО)**, имащи стойности истина (**true**) или лъжа (**false**) (кодират се с двоичните **1** и **0**). Ако АО се именоват те се разглеждат като **логически променливи или константи**.

Разгледаните дотук съждения са прости. Те могат да служат за образуване на сложни съждения при използване на логически връзки (ЛВ). ЛВ дефинират логическите операции. **Съждения свързани със знаците за логически операции образуват логически изрази**.
Функцията, която приема само две стойности и зависи само от двоични променливи, се нарича **логическа функция.**

1.6 Основни логически операции

Основните логически връзки са „И”, „ИЛИ”, „НЕ“.

Основните логически операции са **конюнкция, дизюнкция, отрицание**.

Има и други логически операции (функции), като **изключваща конюнкция**, **изключващо или (**по известна като **сума по модул 2**), **импликация**, **равнозначност** и т.н.

Конюнкцията (логическо умножение) е операция, при която две логически променливи се свързват с **логическата връзка И**. Тя има два аргумента и стойност 0, когато поне един от аргументите й има стойност 0, и стойност 1, когато и двата аргумента са равни на 1. Означава се със знака **∧** или с AND, пр. **A AND B** или **A ∧ B**

Таблица на истинност:

A	B	A ∧ B
0	0	0
0	1	0
1	0	0
1	1	1

Дизюнкцията е операцията, при която две логически променливи се свързват **с логическата връзка ИЛИ**. Тя има два аргумента и има стойност 1, когато поне един от аргументите и има стойност 1, и стойност 0, когато и двата аргумента са 0. Означава се със знака **V** или с OR, например:
A OR B или **A V B**.

Таблица на истинност:

A	B	A V B
0	0	0
0	1	1
1	0	1
1	1	1

Отрицание - Логическо отрицание е операцията, при която се получава нова логическа променлива със стойност, обратна на началната променлива. Отрицанието има един аргумент и променя стойността на аргумента от 1 в 0 или обратно, от 0 в 1. Срещат се различни варианти на означаване**: !, NOT,** ¬**.**

Таблица на истинност:

A	**¬A**
0	**1**
1	**0**

Приоритет на логическите операции:

Най-висок приоритет има отрицанието, следвано от конюнкцията и дизюнкцията

Пример: **X OR Y AND Z**

Ако операциите имат еднакъв приоритет, то те се изпълняват отляво надясно. **X AND Y AND Z**

Приоритет пред всички операции имат операциите, които са в скоби. **(X OR Y) AND Z**

Примери за логически изрази (първо се смятат аритмет. изрази)

Логически израз, проверяващ условието за съществуване на триъгълник със страни A, B и C:

(A + B > C) AND (B + C > A) AND (C + A > B)

Логически израз, проверяващ условието за съществуване на равнобедрен триъгълник.

(A = B) OR (B = C) OR (C = A)

Закони (теореми) на Де Морган:

- NOT (A AND B) = (NOT A) OR (NOT B) <=> ¬(A ∧ B) = ¬A ∨ ¬B,
- NOT (A OR B) = (NOT A) AND (NOT B), <=> ¬(A ∨ B) = ¬A ∧ ¬B.

1.7 Функционално пълна система

Всяка система, съставена от няколко основни логически функции, с помощта на които може да се представят всички останали лог. функции се нарича **функционално пълна**. **Такава функционално пълна система образуват конюнкцията, дизюнкция и отрицание**. С др. думи възможно е да се състави логически израз, описващ произволно сложна двоична функция. Най-удобно това става в така наречените нормални форми.

Използват се две такива :

- **Конюнктивна нормална форма (КНФ)** – конюнкция от дизюнкции;
- **Дизюнктивна нормална форма (ДНФ)** – дизюнкция от конюнкция ;

Пример: В таблица е дадена булевата функция F с нейната таблица на истинност. Да се състави ДНФ на F.

Лесно се забелязва, че F приема стойност 1 когато поне два от входните аргументите са 1. Необходима ни е дизюнкция от конюнкциите на ония комбинации на X_1, X_2 и X_3, за които функцията е единица. Т.е.

$$F_Д = (\neg X_1 \wedge X_2 \wedge X_3) \vee (X_1 \wedge \neg X_2 \wedge X_3) \vee (X_1 \wedge X_2 \wedge \neg X_3) \vee (X_1 \wedge X_2 \wedge X_3)$$

Вход			Изход
X_1	X_2	X_3	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Таблица на истинност за примера от 1.7

1.8 Логически елементи и схеми

Логически елементи - физически елементи, които имат няколко входа и един изход и в съответствие със заложените в тях правила изработват двоичен изходен сигнал чрез преобразуване на съвкупността от подавани на входовете двоични сигнали**.**

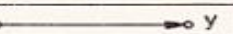
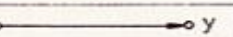
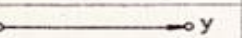
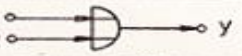
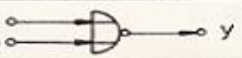
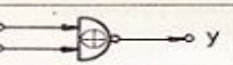
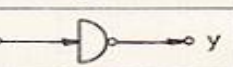
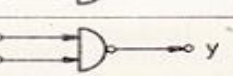
Логическите схеми се градят на свързани по определени правила логически елементи.

n = 2

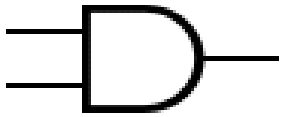
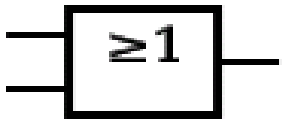
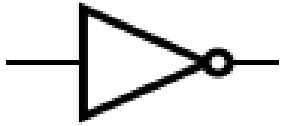
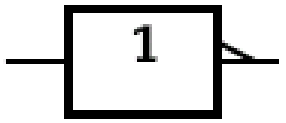
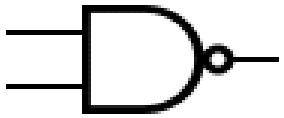
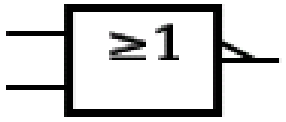
x_1	0	0	1	1	Условно означение	Наименование	A*	Комутативна	Инверсия на
x_2	0	1	0	1					
f_0	0	0	0	0	0	Константа 0	0	–	f_{15}
f_1	0	0	0	1	$x_1\ x_2$ $(x_1 \wedge x_2)$	И – логическо произведение, конюнкция	2	Да	f_{14}
f_2	0	0	1	0	$x_1 \not\leftarrow x_2$	Забрана по x_2	2	Не	f_{13}
f_3	0	0	1	1	x_1	Променлива x_1	1	–	f_{12}
f_4	0	1	0	0	$x_1 \not\rightarrow x_2$	Забрана по x_1	2	Не	f_{11}
f_5	0	1	0	1	x_2	Променлива x_2	1	–	f_{10}
f_6	0	1	1	0	$x_1 \oplus x_2$	Сума по модул 2, неравнозначност	2	Да	f_9
f_7	0	1	1	1	$x_1 \vee x_2$ $(x_1 + x_2)$	ИЛИ – логическа сума, дизюнкция	2	Да	f_8
f_8	1	0	0	0	$x_1 \downarrow x_2$	ИЛИ – НЕ – стрелка на Пирс, инверсия на логич. сума	2	Да	f_7
f_9	1	0	0	1	$x_1 \sim x_2$	Логическа равнозначност	2	Да	f_6
f_{10}	1	0	1	0	$\bar{x}_2$	НЕ, инверсия на x_2	1	–	f_5
f_{11}	1	0	1	1	$x_1 \leftarrow x_2$	Импликация от x_2 към x_1	2	Не	f_4
f_{12}	1	1	0	0	$\bar{x}_1$	НЕ, инверсия на x_1	1	–	f_3
f_{13}	1	1	0	1	$x_1 \rightarrow x_2$	Импликация от x_1 към x_2	2	Не	f_2
f_{14}	1	1	1	0	$x_1 \mid x_2$	И – НЕ – щрих на Шефер, инверсия на логическо произведение	2	Да	f_1
f_{15}	1	1	1	1	1	Константа 1	0	–	f_0

A* – Брой на аргументите, от които зависи съществено.

Фиг. 1.5 Всички логически функции с два аргумента

№	Наименование	Графичен символ	Функция в И,ИЛИ,НЕ
0	Константа 0		$y = 0$
1	Логическо произведение		$y = x_1 x_2$
2	Забрана по x_2		$y = x_1 \bar{x}_2$
3	x_1		$y = x_1$
4	Забрана по x_1		$y = \bar{x}_1 x_2$
5	x_2		$y = x_2$
6	Сума по модул 2		$y = \bar{x}_1 x_2 \vee x_1 \bar{x}_2$
7	Логическа сума		$y = x_1 \vee x_2$
8	Инверсия на логическа сума		$y = \overline{x_1 \vee x_2}$
9	Логическа равнозначност		$y = \bar{x}_1 \bar{x}_2 \vee x_1 x_2$
10	Инверсия на x_2		$y = \bar{x}_2$
11	Импликация от x_2 към x_1		$y = x_1 \vee \bar{x}_2$
12	Инверсия на x_1		$y = \bar{x}_1$
13	Импликация от x_1 към x_2		$y = \bar{x}_1 \vee x_2$
14	Инверсия на логическо произведение		$y = \overline{x_1 x_2}$
15	Константа 1		$y = 1$

Фиг. 1.6 Същите 16 функции реализирани чрез И, ИЛИ, НЕ

AND	A B		A B		A ^ B
OR	A B		A B		A ∨ B
NOT	A		A		¬ A
NAND	A B		A B		¬(A ^ B)
NOR	A B		A B		¬(A ∨B)

Фиг. 1.7 Логически ел. елементи, реализиращи основните лог. операции

Графично логическите елементи се означават с помощта на 2 вида прости блокови схеми:

Първият вид форми, наречени“**отличаващи се**“ са по-традиционни и по-популярни в някои среди. Те са и по-удобни и при чертане на ръка. Често те се определят като"military", като се има предвид, че произхождат от военната област. От своя страна означенията, определени като"**правоъгълни**" са по-общи и могат да се използват за означение на по-голяма група устройства.

Най-евтини за производство са NAND и NOR елементи. **Charles Peirce** доказа, че NAND елементите самостоятелно (както и NOR елементите

сами) могат да бъдат използвани за получаването на всички други логически елементи.

Всяка последователност от връзки от вход на схемата до неин изход се нарича път. Всеки път преминава през определени елементи и може да се опише като последователност от техните номера.

Броят на елементите, през които минава пътят се нарича **стъпалност на пътя** и определя необходимото време за разпространение на сигнала от началото (входа на схемата) до нейния край (изхода на схемата).

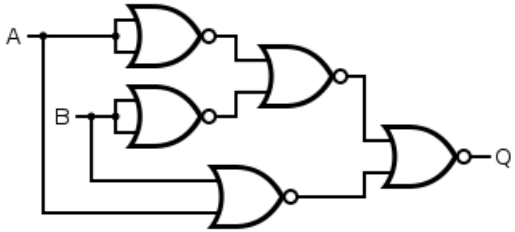
Логическите елементи са основни градивни елементи в компютрите и дискретната автоматика.

Пример за логическа схема на еднобитов суматор с 3 входа.

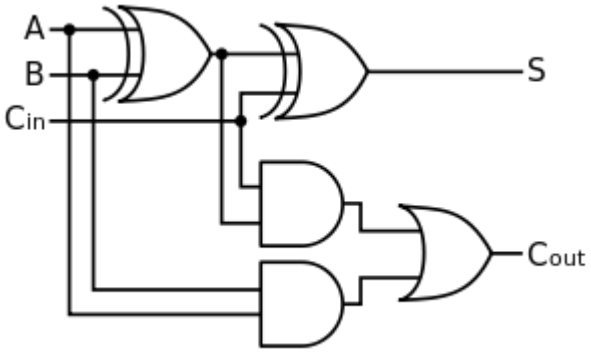
Това е основният елемент в аритметично-логическото устройство на процесора. Входът C_{in} е за преноса от предишния бит. Показаната схема на фиг. 1.10 е тристепенна (тристъпална). За да се сумират две числа в двоична бройна система за всяка съответна двойка битове е необходима една такава схема, като C_{out} от текущия сбор се явява C_{in} за следващия. Един XOR елемент може да бъде реализиран посредством пет NOR елемента по схемата от фиг. 1.9.

Входове			Изходи	
A	**B**	**C_{in}**	**C_{out}**	**S**
0	**0**	**0**	**0**	**0**
0	**0**	**1**	**0**	**1**
0	**1**	**0**	**0**	**1**
0	**1**	**1**	**1**	**0**
1	**0**	**0**	**0**	**1**
1	**0**	**1**	**1**	**0**
1	**1**	**0**	**1**	**0**
1	**1**	**1**	**1**	**1**

Фиг. 1.8 Сума по модул 2 (XOR)



Фиг. 1.9 XOR чрез пет NOR елемента



Фиг. 1.10 Схема на суматор