Доц. д-р инж. Агоп Саркисян

Архитектура и организация на изчислителни системи

Учебно пособие
за провеждане на упражнения по
едноименната дисциплина

Свищов
2003

Предговор

Настоящото пособие е разработено, за да даде възможност на студентите от специалност “Информатика” да използват методически материали при подготовката си по дисциплината “Компютърни архитектури” в упражненията по едноименната дисциплина. То може да бъде и използвано и за дисциплините “Архитектура и организация на изчислителни системи” и “Архитектура и организация на персонални компютри”, които се четат към Свободния факултет на СА “Д.А.Ценов” – Свищов.

В пособието са разгледани теоретическите – аритметични и логически - основи на компютърните системи, както и основните технологии за изготвяне на компютърните компоненти, звена и възли и принципите на тяхното функциониране. Този материал не е включен в основния учебник по споменатите по-горе дисциплини, но е необходим за изясняване и по-добро усвояване на изложените в учебника знания. Дадени са задачи и въпроси за самоподготовка на студентите, както и нагледни графични материали, съдържащи илюстрациите към курсовете по тези дисциплини.

Авторът се надява, че пособието ще послужи за по-задълбоченото изучаване на изчислителните системи, както и при усвояването на базата от необходими понятия и категории, използвани в компютърните науки.

Както всяко първо издание, така и това може да бъде подобрявано и усъвършенствано. Ето защо авторът с благодарност ще приеме всички забележки и предложения за допълнения, които ще бъдат използвани за повишаване на качеството на бъдещите версии на пособието.

СЪДЪРЖАНИЕ

I. Аритметични и логически основи на изчислителните системи

1. Аритметични основи на изчислителните системи.

Езикът на числата си има своя азбука. За да можем да го разбираме е необходимо да я знаем, и за да я използваме и да можем чрез нея да го разбираме. Това означава да се запознаем с основните понятия, използвани в тази област.

1. 1. Бройни системи.

Бройна (числова) система се нарича система от символи и правила за тяхната употреба, посредством които може да се изобрази всяко число. Прието е символите на бройните системи да се наричат цифри. Цифрата е символ, изразяващ цяла величина. Подредените по специални правила на бройната система цифри образуват ***число***. Всяко местоположение, което те могат да заемат вътре в записа при представяне на едно число се нарича ***позиция*** или ***разред,*** като отделните разреди се номерират последователно. Общото означение на цифрите в едно число се дава със символа a_i. Номерирането на разрядите се извършва отдясно наляво, като най-младшият разряд има нулев номер.

Пример:

В числото 561 цифрата 1 е разположена в най-десния (0-левия) разряд, а 5 във втория разряд.

Бройните системи биват два основни вида: ***непозиционни*** и ***позиционни***.

При непозиционните бройни системи количественият еквивалент на всеки символ не зависи от позицията му в числото, а само от неговата собствена стойност.

Пример за непозиционна система е Римската бройна система, която за означаване на цифрите си служи с букви от латинската азбука: I-1, V-5, X-10, L-50, C-100, M-1000. Ако се разгледа числото XXII се вижда, че символът X има количествен еквивалент 10 и в двата случая, независимо от мястото му в числото. Когато по-малък по стойност символ се поставя пред такъв с по-голяма стойност се кодира действието изваждане - например IX: от 10 се изважда 1. Ако такъв символ се постави след по-голям, се кодира действие събиране – например XI → 10+1 = 11.

При позиционните бройни системи значението на дадена цифра зависи не само от нейната собствена стойност, но и от мястото (позицията), което тя заема в числото.

Пример:

Да вземем числата “53” и “35” – в първият случай цифрата “5” означава 5 десетици, а във втория същата цифра означава 5 единици. Тези числа могат да бъдат представени по следния начин:

$53 = 5*10 + 3*1 = 5*10^1 + 3*10^0$

$35 = 3*10 + 5*1 = 3*10^1 + 5*10^0$

Виждаме, че и в двете представяния участва едно и също число – 10, което е поставено на съответна позиция и повдигнато на определена степен. Това число се нарича ***основа на бройната система*** и най-често се означава с ***p***.

Отделните цифри, както беше посочено по-горе се означават с коефициента $\mathbf{a_i}$, където i = 0, 1, 2,…, n.

Тогава общият вид на представяне на цяло число (*N*) в позиционна бройна система е следният:

$$N = a_n p^n + a_{n-1} p^{n-1} + ... + a_1 p^1 + a_0 p^0 ,$$

където: $0 \le a_i \le p\text{-}1$, $p \le 2$, i = 0, 1, 2, 3, …, n

Както се вижда от неравенството, за основата *p* съществуват следните ограничения:

- *p* трябва да е по-голяма или равна на 2. Ако тя е 1, ще се получи бройна система, в която има само един символ за изобразяване на всички числа, а това е невъзможно;

- трябва да бъде цяло число, тъй като основата определя в тълковен смисъл броя на символите (цифрите), с които си служи системата, а той не може да е дробен;

- основата няма ограничение отгоре т.е. може да има множество различни позиционни бройни системи с различни основи, стига да са изпълнени горните две условия.

Освен цели обаче, съществуват още дробни и смесени (с цяла и дробна част) числа. По аналогия с целите числа, без доказателство ще приемем, че общият вид на дробните числа в позиционна бройна система изглежда по следния начин:

$$M = a_{-1} p^{-1} + a_{-2} p^{-2} + ... + a_{-(n-1)} p^{-(n-1)} + a_{-n} p^{-n}$$

където $0 \le a_i \le p\text{-}1$, $p \le 2$, i = 0, 1, 2, 3, ..., n

От цялото множество различни позиционни бройни системи в компютърните системи и човешката практика са намерили най-широко приложение следните видове: **десетична (*p*=10)**, **двоична (*p*=2)**, **осмична (*p*=8)**, **шестнадесетична (*p*=16)**, **двоично-десетична (BCD)**. В таблицата по-долу е показано съответствието на цифрите им:

десетична	***двоична***	***осмична***	***шестнадесетична***	***BCD***
0	0	0	0	0000
1	1	1	1	0001
2	10	2	2	0010
3	11	3	3	0011
4	100	4	4	0100
5	101	5	5	0101
6	110	6	6	0110
7	111	7	7	0111
8	1000	10	8	1000
9	1001	11	9	1001
10	1010	12	A	0001 0000
11	1011	13	B	0001 0001
12	1100	14	C	0001 0010
13	1101	15	D	0001 0011
14	1110	16	E	0001 0100
15	1111	17	F	0001 0101
16	10000	20	10	0001 0110

От таблицата се вижда, че при бройната система с основа p=16 трябва да се използват допълнителни символи освен тези на десетичната система – за такива се приемат латинските букви A, B, C, D, E и F. Другата забележка, която можем да направим, е, че при двоично-десетичната бройна система се използват по 4 двоични разряда (тетрада) за представянето на всяка десетична цифра на числата. Това означава, че за да се изразят десетичните цифри са възможни различни комбинации на двоичните разряди в една тетрада. Броят на тези комбинации е голям – 48048, но се използват само няколко, като най-популярна е **естествената двоично-десетична бройна система** (нарича се още код 8 4 2 1 заради теглата на отделните двоични разряди на тетрадата).

1. 2. Правила за преминаване от една бройна система в друга.

Наличието на различни бройни системи предполага възможността едни и същи числа да могат да се представят по различен начин в тях. Тази възможност се реализира чрез специални правила, произтичащи от общите правила за делимост на числата. Трябва да се отбележи, че за представянето на целите и дробните числа се използват различни правила. Понеже десетичната бройна система има ключова роля в човешката практика, ще разгледаме първо как се преобразуват числа от десетична в p-ична бройна система, а след това от p-ична – в десетична.

- правила за преобразуване (алгоритъм) на <u>цели</u> числа от десетична в различна бройна система – осъществява се в следната последователност:

  1) изходното число се дели на основата на новата система, представена чрез средствата на старата система. При деленето се получава частно или остатък;

  2) проверява се дали частното е по-малко от 0. Ако отговорът е "да", се преминава към 4), ако е "не" – се преминава към 3);

  3) частното се приема за ново делимо и следва връщане към 1);

  4) записват се всички остатъци от делението в обратен ред на получаването им и се получава числото в новата бройна система.

<u>Пример 1:</u> Да се преобразува десетичното число 28 в двоична бройна система, т.е.:

$28_{(10)} \rightarrow ?_{(2)}$

28:2=14	ост. 0	↑
14:2=7	ост. 0	|
7:2=3	ост. 1	|
3:2=1	ост. 1	|
1:2=0	ост. 1	|

Полученото число е **11100**.

Проверката на посочения алгоритъм се извършва чрез **обратно преобразуване** на двоичното число в десетично, като се използва общия вид на целите числа, представени в позиционна бройна система, т.е.:

Проверка: $N = 1*2^4 + 1*2^3 + 1*2^2 + 0*2^1 + 0*2^0 = 16 + 8 + 4 = 28$

Пример 2: Да се преобразува десетичното число 51 в двоичен вид.

$51_{(10)} \rightarrow ?_{(2)}$

51:2=25 ост. 1
25:2=12 ост. 1
12:2=6 ост. 0
6:2=3 ост. 0
3:2=0 ост. 1
1:2=0 ост. 1 Полученото число е **110011**.

Проверка: $N = 1*2^5 + 1*2^4 + 0*2^3 + 0*2^2 + 1*2^1 + 1*2^0 = 32 + 16 + 2 + 1 = 51$

♦ преобразуване на дробни числа от десетична в друга бройна система:

1) изходното десетично дробно число се умножава по основата на новата система, представена със средствата на старата. При умножението се получава цяла и дробна част;

2) цялата част се запомня и следва връщане към 1);

3) проверява се дали е достигната необходимата точност на преобразуване (Δ), която е зададена предварително и определя броя на умноженията. Ако проверката е положителна, се преминава към 4), а ако не е – към 1);

4) заместват се всички цели части на произведенията в реда на тяхното получаване, като се предхождат от запетая и нула.

Пример: Да се преобразува десетичното число 0,23 в двоична бройна система, т.е.:

$0,23_{(10)} \rightarrow ?_{(2)}$ $\Delta = p^{-4}$ - определя броя на умноженията

0,23*2=0,46
0,46*2=0,96
0,96*2=1,84
0,84*2=1,68 => 0,0011

Проверката отново се извършва чрез **обратно преобразуване** на двоичното число в десетично, като се използва общия вид на дробните числа, представени в позиционна бройна система, т.е.:

Проверка: $N = 0*2^{-1} + 0*2^{-2} + 1*2^{-3} + 1*2^{-4} = 1/8 + 1/16 = 3/16 = 0,1874$

Вижда се, че се получава разминаване в стойностите поради зададената точност на преобразуване – колкото е по-голяма, толкова то е по-точно.

Примери за превръщане на цели и дробни числа от десетична в двоична бройна система:

1) $25_{(10)} \rightarrow 11001_{(2)}$
25:2=12 ост. 1
12:2=6 ост. 0
6:2=3 ост. 0
3:2=1 ост. 1
1:2=0 ост. 1

2) $0,25_{(10)} \rightarrow 0,01_{(2)}$
0,25*2=0,5
0,5*2=1,0

3) $25_{(10)} \rightarrow 221_{(3)}$
25:3=8 ост. 1
8:3=2 ост. 2
2:3=0 ост. 2

4) $0,452_{(10)} \rightarrow 0,21122_{(5)}$ $\Delta = p^{-5}$
0,452*5=2,26
0,26*5=1,3
0,3*5=1,5
0,5*5=2,5
0,5*5=2,5

Превръщането на числа от различни бройни системи, които имат връзка на основите от типа **p = q** $^{\mathbf{n}}$ е много лесно, защото не се преминава през десетична система.

Например, за да се преобразува числото 10011011 от двоична в осмична бройна система, то се разделя на тройки (защото $8=2^3$) отдясно наляво и всяка тройка се преобразува поотделно в осмична цифра:

$10,011,011_{(2)} = 2,3,3 = 233_{(8)}$

Ако същото число трябва да се преобразува в шестнайсетична система, то се разделя по същия начин на четворки (тетради):

$0,1001,1011_{(2)} = 0,9,B = 9B_{(16)}$

Както вече беше казано, при двоичната бройна система основата е 2 (p = 2). Тогава общият вид на представяне на числата в двоичен код е следния:

$$N = a_n 2^n + a_{n-1} 2^{n-1} + ... + a_1 2^1 + a_0 2^0$$

Тъй като двоичната система заема много важно място в изчисленията в компютърните системи, първо ще разгледаме по-подробно начина, по който се извършват те, т. нар. двоична аритметика.

Двоичната аритметика се базира на следните елементарни таблици за събиране (I), изваждане (II) и умножение (III):

I	II	III
0 + 0 = 0	0 - 0 = 0	0 * 0 = 0
0 + 1 = 1	1 - 0 = 1	1 * 0 = 0
1 + 0 = 1	1 - 1 = 0	0 * 1 = 0
1 + 1 = 0 (1)	10 - 0 = 1	1 * 1 = 1

Основните действия, които могат да се осъществят с двоичните числа, както и при числата в другите бройни системи, са: събиране, изваждане, умножение и делене. По-долу са дадени примери за извършването им:

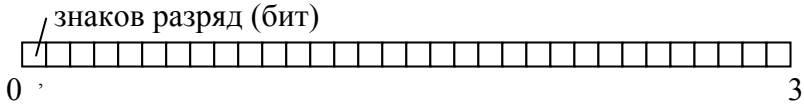
```
  1
1 10 1 1 1                1 1  1
  101111            10010010111          101                  101
+  11011          -      101001        * 110                * 110
   11101            10001101110          000 \             / 101
 1100111                               +  101  } частични {+  101
                                          101 /  произведения \ 000
  11011101:101=101100,001                11110                11110
- 101
  00111
   - 101
     0101
    - 101
     00001000
         - 101
           011  ….и т.н.
```

Умножението представлява събиране с изместване на множимото (чрез получаване на т. нар. *частични произведения*) наляво или надясно, според това с кои разряди на множителя се започва умножението – младшите или старшите. Когато се налага да се изважда 1 от разряд, в който има 0, се взема “на заем” от съседните по-старши разряди, като “заемът” се разпределя по следния начин – на 0-левия разряд се присвоява 10, а на междинните нулеви разряди – единици.

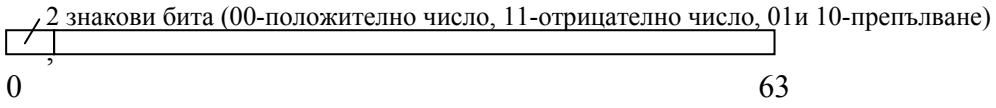
1. 3. Форми на представяне на числата в изчислителните системи.

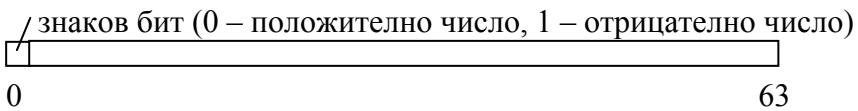
В компютърните системи се използват две форми на представяне на числата: **с фиксирана запетая** и **с плаваща запетая**.

а) формата на представяне с фиксирана запетая (т. нар. *естествена форма*) може да се илюстрира чрез графично представяне на клетка от паметта на компютърната система (32-битова), съдържаща такова число, т.е.:



Първият разряд (бит) не е значещ, а се приема за знаков. Ако той има стойност 0, числото е положително; ако е 1 – то е отрицателно. Има два вида компютърни системи – при едните най-левият разряд на клетката е знаков, а при другите такъв е най-десният. В първия случай числата се третират като дробни, а при втория – само като цели. Във връзка с някои проблеми (препълване на разрядната мрежа) при изпълнение на аритметични операции с фиксирана запетая в съвременните компютърни системи се използват по два знакови разряда и двойна точност:





Примери:

$+53_{(10)} = +110101_{(2)}$

0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	1

$-15_{(10)} = -1111_{(2)}$

1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	1

Представимите по естествен начин числа могат да бъдат в диапазона от 2^{-31} до $1-2^{31}$ или -2^{-63} до $1-2^{63}$. С такава форма на представяне се работи, когато се пресмятат задачи в области, където числата не се различават много в броя на разрядите си. Съществуват обаче области, където едновременно трябва да се работи с много малки и много големи числа – например ядрените изследвания. Ако там се използва естествената форма, най-малките или най-големите числа няма да се поберат в разрядната мрежа на системата. За такива изчисления се използва другата форма на представяне, а именно:

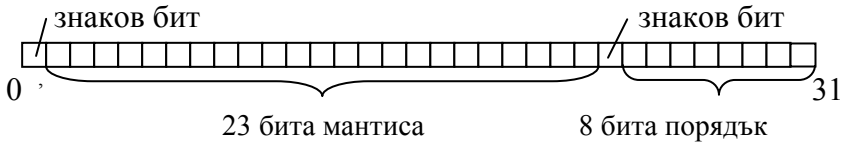
б) форма с плаваща запетая (нормална, полулогаритмична форма) – *нормалната форма* на представяне на числата може да се даде със следната формула:

$N = \pm M * 2^{\pm P}$ където M се нарича мантиса

N е представяното число

p е порядък (характеристика)

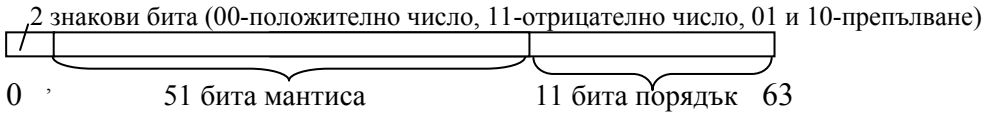
Графично клетката съдържаща число с нормалната форма на представяне има следния вид (при клетка от 32 разряда):

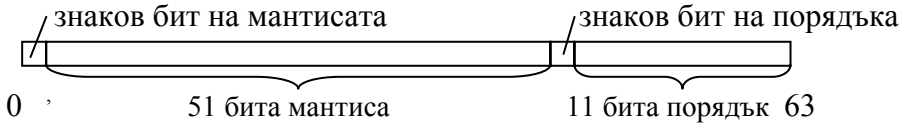


Знакът на мантисата определя знака на числото - положителното число има знаков бит 0, а отрицателното - 1. От знака на порядъка (характеристиката) се определя дали числото е цяло (0) или дробно (1). Представимите чрез тази форма числа заемат диапазона от $-(1-2^{-24})*2^{-128}$ до $(1-2^{-24})*2^{-128}$.

И тук както при фиксирана запетая има малки числа, които излизат извън разрядната мрежа – т. нар. машинни нули. Но при плаваща запетая те са в много по-тесен диапазон - от $-0,5*2^{-128}$ до $0,5*2^{-128}$.

При плаващата запетая също може да се използват два разряда за знака на мантисата и порядъка (при 64-разрядна мрежа):





Обикновено в изчислителните системи, когато диапазонът на представимите числа не достига, преминаването от едната в другата форма на представяне става автоматично.

Пример: Двоичното число 0,101111 може да се представи така[1]:

0,1011 **0** 101 = $11*2^5$ – цяло положително число

0,1011 **1** 101 = $11*2^{-5}$ – дробно положително число

Примери (графично представяне):

$+1011,011 = 0,1011011*10^{+100}$

0	0	0	0	1	0	1	1	0	1	1	0	0	1	0	0

$-100110,1011 = -0,1001101011*10^{+110}$

1	1	0	0	1	1	0	1	0	1	1	0	0	1	1	0

По принцип всички изчислителни системи могат да работят и с двете форми на представяне.

Двоично-десетичните числа (BCD) имат специален вид на представяне – пакетиран и зонов формат. При пакетираният формат в 1 байт се записват две BCD-цифри: $25_{(10)} \rightarrow 0010\ 0101_{(BCD)}$ – двете цифри са 0010 и 0101.

За представяне на двоично-десетичните числа в зонов формат са нужни два байта. В първия се записва знака на числото и първата цифра, а във втория отново знака и втората цифра:

$25_{(10)}$ се представя като 0000 0010
0000 0101

[1] Знаковите разряди на мантисата и порядъка са почернени.

Тази форма на представяне се използва предимно в дисковите носители.

Естествената, нормалната и формите на представяне на двоично-десетичните числа оказват влияние на начините за осъществяване на аритметични операции в изчислителните системи. В съответствие с тези три форми различаваме **операции с фиксирана запетая**, **операции с плаваща запетая** и **двоично-десетични операции** (BCD аритметика).

1. 4. Аритметични действия с числа, представени в естествена форма (с фиксирана запетая).

При операциите с фиксирана запетая се изпълняват всички аритметични действия – събиране, изваждане, умножение и делене. Но тъй като в компютъра не могат да се правят отделни устройства за четирите действия, в аритметиката е предвидено всички операции да се свеждат до една – събиране, която да се реализира само от едно устройство.

♦ **събиране с фиксирана запетая** – при него се събират разрядите на числата, а знакът на новото число (сумата) се определя според това кое от събираемите е по-голямо по абсолютна стойност (по модул).

Например: x = 0,011 (+3)
y = 0,010 (+2)
x+y = 0,101 (+5)

Примери за събиране на числа (когато $|x+y| \leq 0$):

1) x = +0,0101
y = +0,1001
x+y = +0,1110

2) x = -0,0101
y = -0,1001
$x_д$[2] = 1,1011
$y_д$ = 1,0111
$x_д+y_д$ = 1,0010
+1
x+y = -0,0011

При събирането на двоични числа с еднакви знаци обаче, се получава проблем, когато абсолютната сума на $|x+y| \geq 1$. Тогава става прехвърляне (пренос) от цифрите на числата в знаковия разряд и се получава неверен резултат – при събиране на положителни числа се получава отрицателна сума, а при събиране на отрицателни числа – положителна. Ситуацията се нарича *препълване на разрядната мрежа* (overflow). Например:

x = 0,101 (+5)
y = 0,110 (+6) - при събиране на числа > 0
x+y = 1,011 (-3)

x = -0,101 (-5)
y = -0,110 (-6)
$x_д$ = 1,011 (-5)
$y_д$ = 1,010 (-6) - при събиране на числа < 0
$x_д+y_д$ = 10,101 (+5)
└ този разряд отпада, тъй като в машината има само един разряд за знака

Когато се получи грешка (препълване), тя трябва да се идентифицира по някакъв начин (тъй като получения резултат е неверен) и евентуално да се изпълни действие за коригиране на грешката, ако това е възможно. За да се

[2] Индексът "д" се използва за означаване на допълнителния код на числата, който се разглежда малко по-долу при изваждане на числата с фиксирана запетая.

получи такъв индикатор се въвеждат *модификационни кодове*, които се състоят от два разряда за знака. Тогава положителното число +3 се представят в двоичен код като 00,011, а -3 като 11,011. Ако при събирането в тези два знака се получат комбинации 01 или 10, това е сигурен индикатор, че е настъпило препълване.

Препълването може да се коригира като се премине от една форма на представяне на числата към друга – т.е. от фиксирана запетая към плаваща. При това действие мантисата на едно от събираемите се измества надясно и тогава се извършва събирането в плаваща запетая, което се разглежда в следващата точка на пособието. Намаляването на мантисата по този начин води и до промяна и на порядъка.

Тогава дадените по-горе примери за събиране (при $|x+y| \geq 0$) могат да се доразвият чрез използване на модифицирани кодове ($x_м$ и $y_м$) по следния начин:

1) $x = 0,101$ $x_м = 00,101$
$y = 0,110$ $y_м = \underline{00,110}$
$(x+y)_м = \underline{01},011$ - Препълване!

2) $x = -0,101$ $x_{дм} = 11,011$
$y = -0,110$ $y_{дм} = \underline{11,010}$
$(x_д+y_д)_м = 1\underline{10},101$ - Препълване!

♦ **изваждане с фиксирана запетая** – то се свежда до действието събиране съгласно следната формула: $x - y = x + (-y)$, т.е. представлява събиране на числа с различни знаци. При изваждането се използват различни методи за автоматично определяне знака на сумата. Най-разпространено е използването на **аритметично допълнение** (допълнителен код). Този код се получава по следния начин: за положителните числа той съвпада със самото число, а за отрицателните нулите на числото се заменят с единици, единиците с нули, а към най-десния (най-младшия) числов разряд се прибавя 1. След като се съберат двете числа получената сума отново се преобразува от допълнителен в прав код като се заменят нулите с единици, единиците с нули и към най-младшия разряд на числото се прибави 1.

Например, ако искаме да извадим числата 0,011 (+3) и 0,101 (+5), използваме формулата $x – y = x + (-y)$ и процедираме по следния начин:

$x = 0,011$ $x_д = 0,011$ $x+y = 0,001$
$y = -0,101$ $y_д = \underline{1,011}$ $\underline{+1}$
$y_д = 1,010$ $x_д+y_д = 1,110$ $0,010$
$\underline{+1}$
$1,011$

Тогава за правия код на разликата се получава -0,010, т. е. числото -2 в двоичен вид.

Примери за изваждане:

1) $x = –0,0101$ $x_д = 1,1011$
$y = +0,1001$ $y_д = \underline{0,1001}$
$x_д+y_д = 0,0100$

2) $x = +0,0101$ $x_д = 0,0101$
$y = –0,1001$ $y_д = \underline{1,0111}$
$x_д+y_д = 1,1100$ или в прав код -0,0100

♦ **умножение с фиксирана запетая** – при може да се умножава с младшите или старшите разряди на множителя:

Примери:

1) x=00,101 y=00,111

```
  101
 *111
 ---
  101
+ 101
 ---
 101
---
100011
```

x*y=00,100011

2) x=00,101 y=11,101

```
  101
 *101
 ---
  101
+101
 ---
11001
```

x*y=11,11001

Знакът на произведението се определя чрез използване на определена логическа функция (сума по модул 2).

♦ **делене с фиксирана запетая** – докато умножението може да се реализира като събиране с изместване на множимото, при деленето това не е така. Деленето на две двоични числа се извършва по специални алгоритми – един от тях е чрез последователно изваждане на делителя от делимото с възстановяване на остатъка, за да се определи колко пъти делителят се съдържа в поредната част от делимото. Процедира се по следния ред:

1. Изважда се делителя от делимото с използване на допълнителен код (т.е. действие изваждане се свежда до събиране). Ако знакът на разликата е положителен в поредната цифра на частното се записва 1 и изваждането продължава за определяне на следващата цифра на частното;

2. Ако знакът на разликата е отрицателен (11 при модифициран код в знаковите разряди), това означава че делителят повече не се съдържа в делимото и в поредната цифра на частното се записва 0. Преминава се към т.3;

3. Възстановява се остатъкът като към отрицателната разлика се прибавя делителя в прав код;

4. Към възстановения остатък се прибавя (смъква) поредната цифра от делимото и операцията по изваждането се повтаря;

5. Това продължава докато се получат всички цифри на частното, т.е. до зададената точност или разрядност.[3]

Пример: Да се разделят двоичните числа в модифицирани кодове 00,10111 и 00,11101.

x=00,10111
y=00,11101
x:y=00,10100

```
 00,10111:00,11101=00,10100
+11,00011
 11,11010   } възстановяване на
+00,11101   }      остатъка
 00,10111  ← изместване
  01,01110
 +11,00011
  00,10001
 +11,00011
  11,10100   } възстановяване на
 +00,11101   }      остатъка
  00,10001←
   01,00010
  +11,00011
   00,00101
  +11,00011
   11,01000   } възстановяване на
  +00,11101   }      остатъка
   00,00101←
    00,01010
   +11,00011
    11,01101
```

[3] Забележка: Понеже числата са представени с фиксирана запетая и се третират като дробни, то винаги при делението делителят е по-голямо по абсолютна стойност число от делимото, за да се получи частно – дробно число. В противен случай ще се получи препълване.

1. 5. Аритметични действия с числа, представени в нормална форма (с плаваща запетая).

Както вече казахме, числата във формат плаваща запетая се представят в следния общ вид:

$x = \pm X_M * B^{\pm X_E}$, където X_M, Y_M – мантиси

$y = \pm Y_M * B^{\pm Y_E}$ B –основа на бройната система

X_E, Y_E - порядък

При работа с плаваща запетая правилата за извършване на аритметични действия произтичат от правилата за работа със степени. Най-сложни и бавни са операциите събиране и изваждане, защото могат да се събират и изваждат само числа с еднакви степенни показатели и с еднаква основа. Затова предварително степенните показатели се изравняват към по-големия от двата (основата е една и съща – p=2).

Използват се формулите:

$$\left.\begin{aligned} x + y &= \left(\pm X_M * B^{X_E - Y_E} + \left(\pm Y_M\right)\right) * B^{Y_E} \\ x - y &= \left(\pm X_M * B^{X_E - Y_E} - \left(\pm Y_M\right)\right) * B^{Y_E} \end{aligned}\right\} \text{ при } X_E \le Y_E$$

Пример:

x = 00,1101 0 101 x + y = ?

y = 00,110 0 110

x = 00,01101 0 110 - изравняване на степенните показатели и

y = 00,110 0 110 преместване на мантисата на първото число

x+y = 01,00101 0 110

Резултатът е не-нормализирано число (първият разряд след знаковия не е 1). След нормализация (преместване на мантисата на един разряд наляво и намаляване на порядъка с 1) се получава нормализираното число:

x+y = 00,100101 0 110

По аналогичен начин се извършва и изваждането.

За умножение и деление се използват дадените по-долу формули:

$$x * y = \left(\pm X_M * \pm Y_M\right) * B^{(\pm X_E \pm Y_E)}$$

$$\frac{x}{y} = \frac{\pm X_M}{\pm Y_M} * B^{(\pm X_E \mp Y_E)}$$

От формулите се вижда следното: при умножение мантисите на двете числа се умножават, а порядъците се събират; при деление мантисите на двете числа се делят, а порядъците се изваждат; при степенуване порядъците се умножават. Умножението и делението на мантисите се извършва по алгоритмите за умножение и деление на числа с фиксирана запетая. След извършване на действията и тук се извършва нормализация.

Пример:

x = 00,1001 0 110

y = 00,1010 0 100

x*y = (00,1001*00,1010)*10 $^{(110+100)}$ = 00,1011010*10 ^{1010}

1. 6. Аритметични операции с двоично-десетични числа (десетична аритметика).

При изпълнение на аритметичните действия двоично-десетичните числа се представят като цели - със запетая, фиксирана от дясно. При операциите събиране и изваждане те се изравняват по дясната граница, определяна от тази запетая.

Тези операции започват с анализ на знаците. Ако те са еднакви, числата се събират в прав код. Ако са различни, от по-голямото число се изважда по-малкото. Последното (в случая отрицателно число) се представя в допълнителен код и изваждането се свежда до събиране. При това обаче трябва да се отчита следното: представените чрез тетради десетични числа при събирането си трябва да отчитат правилата и на двоичната и на десетичната бройни системи. Това означава, че трябва да се отчита десетичния пренос, когато в един десетичен разряд сумата надрасне 10.

Примери:

1) x = 0010 0101 (25)
   y = 0010 0100 (24)
   x+y = 0100 1001 (49)

2) x = 0010 0110 (26)
   y = 0010 0101 (25)
   x+y = 0100 1011 (411)

При 1) няма десетичен пренос и полученият резултат е верен. Невернният резултат от 2) се получава от неотчитане на десетичния пренос в разряда (получила се е сума по-голяма от 10). За се извърши преноса се налага коригиране на сумата с числото 6 (0110), т.е.:

x = 0100 1011 (411)
y = 0000 0110 (6)
x+y = 0101 0001 (51) – което е верният резултат

До същия резултат може да се достигне, ако първоначално едното от числата се представи в код с излишък 6, т.е. ако му се прибави 6 преди събирането. Ако при изпълнение на събирането по този начин възниква пренос от дадена тетрада към друга, резултатът в този разряд се получава верен. Ако няма преноси в някакви тетради, то за да се получи верен резултат от тях трябва да с извади 6 или да се прибави 1010 (допълнение на 6 до 16).

Пример:

x = 0000 0111 (7) – в код с излишък 6 в двете тетради и числото се
y = 0001 1000 (18) променя на 1010 1101

x = 1010 1101
y = 0001 1000
x+y = 1000 0101 - понеже има пренос само от втората тетрада, първата трябва да бъде коригирана с прибавяне на 1010, т.е.:

x = 1000 0101
y = 1010 0000
x+y = 0010 0101 (25)

В компютъра така описаните корекции се извършват с *универсален десетичен коректор* – устройство, което е част от централния процесор на машината (блока за десетична аритметика).

Пример 2: Да се съберат числата 82 (+1000 0010) и –75 (-0111 0101). Допълнението на 75 до 100 е 25, тогава:

x = 1000 0010 (82)

y = 1000 1011 (25 с излишък 6)

x+y = 0000 1101 – във втората тетрада няма пренос и трябва корекция, т.е.:

\+ 1010

x+y = 0000 0111 (7)

Преносите при корекцията не се отчитат.

Действията умножение и деление с двоично-десетични числа са по-сложни и няма да се разглеждат в рамките на това пособие.

ЗАДАЧИ ЗА САМОСТОЯТЕЛНА РАБОТА

1) $10111011_{(2)} \rightarrow ?_{(10)}$
2) $0{,}1001101_{(2)} \rightarrow ?_{(10)}$
3) $345_{(6)} \rightarrow ?_{(10)}$
4) $0{,}212_{(3)} \rightarrow ?_{(10)}$
5) $765_{(8)} \rightarrow ?_{(2)}$
6) $5CF2_{(16)} \rightarrow ?_{(2)}$
7) $10101110_{(2)} \rightarrow ?_{(8)}$
8) $11111110_{(2)} \rightarrow ?_{(16)}$
9) Да се съберат двоичните числа 110111, 111001, 101011, 111110 с проверка в десетична система.
10) Да се извадят двоичните числа 10011001 и 110101.
11) Да се умножат двоичните числа 1011 и 11011 със старшите и младши разряди на множителя.
12) Да се разделят двоичните числа 11011 и 110.
13) Да се съберат двоичните числа с фиксирана запетая 00,0101 и 00,0011 с използване на прав и допълнителен код.
14) Да се съберат двоичните числа с фиксирана запетая -00,0101 и 00,0011 с използване на прав и допълнителен код.
15) Да се съберат двоичните числа с фиксирана запетая 00,0101 и -00,0011 с използване на прав и допълнителен код.
16) Да се разделят двоичните числа 00,0100 и 00,1010 по алгоритъма с възстановяване на остатъка.
17) Да се съберат двоичните числа с плаваща запетая 00,1010 0 110 и 00,0011 0 100.
18) Да се умножат двоичните числа с плаваща запетая 00,1010 0 110 и 00,0011 0 100.
19) Да се разделят двоичните числа с плаваща запетая 00,1010 0 110 и 00,0011 0 100.
20) Да се съберат двоично-десетичните числа 0010 0010 (22) и 0011 0011 (33).

2. Логически основи на изчислителните системи.

2. 1. Основни понятия в Булевата алгебра.

Логическите основи на изчислителните системи почиват на **булевата алгебра**. Всяка мисъл или изречение, която може да бъде вярна ли невярна, наричаме *съждение*. Към съжденията не се отнасят изреченията или мислите с модален характер (изразяващи молба, заповед, възможност). Съжденията George Bool нарекъл **двоични променливи**. Прието е те да се означават с главните букви от латинската азбука. Съжденията имат вярностна стойност – могат да истина или лъжа – и това се означава като: A = 1 – вярно съждение, A = 0 – невярно съждение. Знакът за равенство се чете "еквивалентно на".

Двоичната функция е функция на n двоични променливи и представлява сложно съждение:

$y = f(x_1, x_2, ..., x_i, ..., x_n)$, където

x_i - прости съждения, двоични променливи
y – сложно съждение
i = 1, 2, 3, 4, …, n

Броят от възможните комбинации между двоичните променливи се наричат набори на логическата функция и се означават с *N*. Очевидно е, че $N = 2^n$. Тогава ако с *F* се означи броя на двоичните функции, за *N* набора за броя на функциите се получава: $F = 2^N$

Например: За функцията с една променлива $y = f(x_1)$ имаме:

$$N = 2^1 = 2 \qquad F = 2^2 = 4$$

За таблично представяне на двоичните функции се използват Таблици на истинността. Четирите функции на една променлива *x* имат следните имена:

x_1	f_0	f_1	f_2	f_3
0	0	0	1	1
1	0	1	0	1

постоянна 0 ($f_0 = 0$) променлива x ($f_1 = x_1$)
променлива $\bar{x}$ ($f_2 = \bar{x}_1$) постоянна 1 ($f_3 = 1$)

2. 2. Булеви функции на две променливи.

Когато имаме функция на две променливи от вида $\boldsymbol{y = f(x_1, x_2)}$, броят на наборите е $N = 2^2 = 4$, а на функциите - $F = 2^4 = 16$. Представени в Таблица на истинност тези функции са следните:

x_1	x_2	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Всички *f* от Таблицата имат наименования и те са:

- постоянна 0 ($f_0 = 0$);
- логическо произведение, конюнкция ($f_1 = x_1 \wedge x_2$);

- импликация ($f_2 = x_2 \rightarrow x_1$; $f_4 = x_1 \rightarrow x_2$);
- променлива x ($f_3 = x_1$; $f_5 = x_2$);
- логическа неравнозначност, сума по модул 2 ($f_6 = x_1 \oplus x_2$);
- логически сбор, дизюнкция ($f_7 = x_1 \vee x_2$);
- "стрелка на Пирс" ($f_8 = x_1 \downarrow x_2$);
- логическа равнозначност ($f_9 = x_1 \otimes x_2$);
- променлива $\bar{x}$ ($f_{10} = \bar{x}_2$; $f_{12} = \bar{x}_1$);
- обратна импликация ($f_{11} = x_2 \leftarrow x_1$; $f_{13} = x_1 \leftarrow x_2$);
- "щрих на Шефер" ($f_{14} = x_1|x_2$);
- постоянна 1 ($f_{15} = 1$).

2. 3. Принципи на суперпозицията – форми на логическите функции.

Джордж Бул е формулирал теорема за суперпозицията (заместване на логическите функции една с друга), според която произволна логическа функция на n променливи може да бъде изразена с помощта на специален набор от логически функции на 2 променливи. Такъв набор се нарича **пълна система от логически функции**. Има различни пълни системи, но най-използваните са:

- класическа база – конюнкция, дизюнкция и отрицание ($\wedge$, $\vee$, $\bar{}$);
- "стрелка на Пирс" ($\downarrow$);
- "щрих на Шефер" (|).

При представянето (заместване) на произволна на логическа функция се използват Таблиците на истинност. За целта Бул предложил да се използват трите логически функции на 2 променливи, наречени **класически базис** – дизюнкция, конюнкция и отрицание.

Процедурата за изразяване на произволна логическа функция чрез класическия базис се състои в следното:

$\Rightarrow$ функцията се задава в табличен вид (наборите на аргументите също са отбелязани);

$\Rightarrow$ намират се онези набори на аргументите, за които функцията е вярна (f=1) / намират се онези набори на аргументите, за които функцията не е вярна (f=0);

$\Rightarrow$ съставят се конституентите на 1 (аргументите образуват конюнкции, като ако аргументът в таблицата е 0 се записва с отрицание, а ако е 1 се записва като самия аргумент) / съставят се конституентите на 0 (аргументите образуват дизюнкции, като ако аргументът е 0 се записва като самия аргумент, а ако е 1 се записва с отрицание);

$\Rightarrow$ съставя се **пълна нормална дизюнктивна форма** (ПНДФ) на функцията (всички конституенти на 1 се обединяват с една обща дизюнкция) / съставя се **пълна нормална конюнктивна форма** (ПНКФ) на функцията (всички конституенти на 0 се обединяват с една обща конюнкция).

Например: Да се намери пълната нормална дизюнктивна форма на функция с три аргумента x_1, x_2 и x_3, зададена със следната таблица на истинност:

$y = f(x_1, x_2, x_3)$ $\quad N = 8$

x_1	x_2	x_3	f	Const 1	Const 0
0	0	0	0		$x_1 + x_2 + x_3$
0	0	1	1	$\overline{x_1} * \overline{x_2} * x_3$	
0	1	0	1	$\overline{x_1} * x_2 * \overline{x_3}$	
0	1	1	0		$x_1 + \overline{x_2} + \overline{x_3}$
1	0	0	1	$x_1 * \overline{x_2} * \overline{x_3}$	
1	0	1	0		$\overline{x_1} + x_2 + \overline{x_3}$
1	1	0	0		$\overline{x_1} + \overline{x_2} + x_3$
1	1	1	0		$\overline{x_1} + \overline{x_2} + \overline{x_3}$

$$f_{\text{пндф}} = (\overline{x_1} * \overline{x_2} * x_3) + (\overline{x_1} * x_2 * \overline{x_3}) + (x_1 * \overline{x_2} * \overline{x_3})$$

$$f_{\text{пнкф}} = (x_1 + x_2 + x_3) * (x_1 + \overline{x_2} + \overline{x_3}) * (\overline{x_1} + x_2 + \overline{x_3}) * (\overline{x_1} + \overline{x_2} + x_3) * (\overline{x_1} + \overline{x_2} + \overline{x_3})$$

Пълните нормални форми на функциите представляват техните логически уравнения – т.е. представени са в аналитичен, а не в табличен вид. Тези уравнения могат да се преобразуват и опростяват подобно на уравненията в обикновената алгебра. За целта обаче е необходимо да се знаят правилата, (законите), по които може да става това. По подразбиране се приема, че дизюнкцията се изразява с +, а конюнкцията с *.

Закони на булевата алгебра:

Комутативен закон – закон за разместването

$$\begin{cases} x_1+x_2 = x_2+x_1 \\ x_1*x_2 = x_2*x_1 \end{cases}$$

Асоциативен закон – събирателен

$$\begin{cases} x_1+(x_2+x_3) = x_1+x_2+x_3 \\ x_1*(x_2*x_3) = x_1*x_2*x_3 \end{cases}$$

Дистрибутивни закони- разпределителни

$$\begin{cases} x_1+(x_2*x_3) = (x_1+x_2)*(x_1+x_3) \\ x_1*(x_2+x_3) = (x_1*x_2)+(x_1*x_3) \end{cases}$$

Закони на тавтологията

$$\begin{cases} x_1*x_2*x_3*\ldots*x_n = x_1 \\ x_1+x_2+x_3+\ldots+x_n = x_1 \end{cases}$$

$$\begin{cases} x_1+(x_1* x_2) = x_1 \\ x_1*(x_1+x_2) = x_1 \end{cases}$$

$$\begin{cases} x_1 * \overline{x_1} = 0 \\ x_1 + \overline{x_1} = 1 \end{cases}$$

Теореми на Де Морган

$$\begin{cases} \overline{x_1 * x_2} = \overline{x_1} + \overline{x_2} \\ \overline{x_1 + x_2} = \overline{x_1} * \overline{x_2} \end{cases}$$

Закон за двойното отрицание

$$\left(\overline{\overline{x}}\right) = x$$

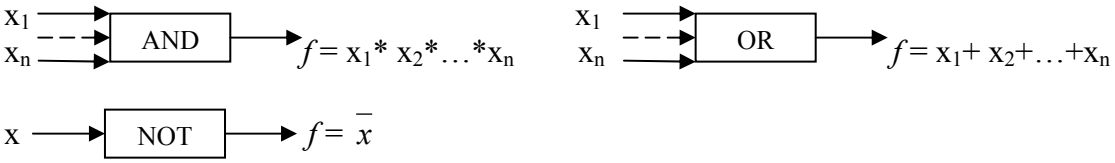
Опростяването на формите на логическите функции с помощта на горепосочените закони се нарича **минимизация.** Получените след минимизацията форми се наричат съответно **минимална дизюнктивна** и **минимална конюнктивна** форми на логическата функция.

Пример за минимизация:

Нека е дадено логическото уравнение $S_{(x1,\ x2)\ пндф} = (\overline{x_1} * x_2) + (x_1 * \overline{x_2})$. За да минимизираме уравнението прибавяме две нули и се получава:

$$S = (\overline{x_1} * x_2)+(x_1 * \overline{x_2}) = (\overline{x_1} * x_2)+(x_1 * \overline{x_2})+0+0 = (\overline{x_1} * x_2)+(x_1 * \overline{x_2})+(x_1 * \overline{x_1})+(x_2 * \overline{x_2}) = \overline{x_1}(x_1 + x_2) + \overline{x_2}(x_1 + x_2) = (x_1 + x_2)(\overline{x_1} + \overline{x_2}) = (x_1 + x_2)(\overline{x_1 * x_2})$$

Конюнкцията, дизюнкцията и отрицанието имат технически аналози, наречени логически елементи. Техническият аналог на конюнкцията се нарича "**и**" (AND), на дизюнкцията се нарича "**или**" (OR), а на отрицанието "**не**" (NOT).

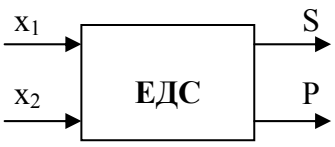


"Стрелката на Пирс" и "щрихът на Шефер" са другите пълни системи логически функции. Първата е обратна на конюнкцията и елементите, които я реализират, се наричат "**не-и**" – NAND. "Щрихът на Шефер" пък е обратен на дизюнкцията и елементите от него се наричат "**не-или**" – NOR. Устройствата в компютърната система могат да се реализират и с двата вида схеми.

Посредством показаните технически аналози на основните логически функции може да се направи технически аналог на логическо уравнение. Такъв аналог се нарича **логическа схема**.

2. 4. Синтез на логически устройства.

Създаването на логически схеми от логически елементи се нарича **синтез на логически устройства**. Пример за синтез е създаването на устройство, сумиращо два разряда от едно двоично число, което се нарича *едноразряден двоичен суматор с два входа*. Чрез използване на кибернетичния принцип на "черната кутия" можем да посочим най-общата му блокова схема:



S – сума
P - пренос

Таблицата на истинност на логическата функция на такъв суматор (съгласно правилата за събиране на двоични числа) е:

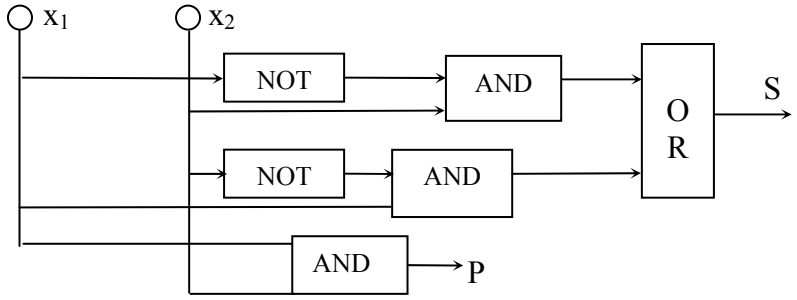
0+0=0
0+1=1
1+0=1
1+1=0 и пренос 1

x_1	x_2	S	P	S const	P const
0	0	0	0		
0	1	1	0	$\overline{x_1} * x_2$	
1	0	1	0	$x_1 * \overline{x_2}$	
1	1	0	1		$x_1 * x_2$

Тогава логическите уравнения на сумата и преноса ще са:

$$S_{(x1, x2)\ пндф} = (\overline{x_1} * x_2) + (x_1 * \overline{x_2})$$
$$P_{(x1, x2)\ пндф} = x_1 * x_2$$

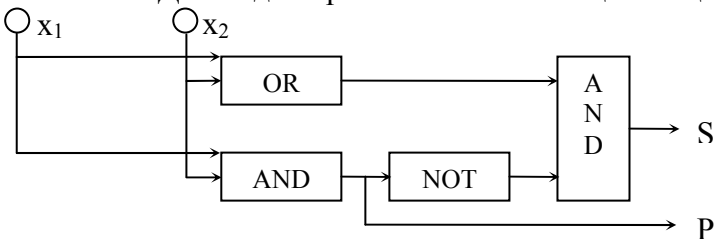
Ако се "нарисува" схемата така описания ЕДС с входове x_1 и x_2, ще видим, че в нея има два елемента "не", три елемента "и" и един "или", т. е.:



Ако се направи минимизация на уравнението на сумата, ще се получи:

$S = (\overline{x_1} * x_2)+(x_1 * \overline{x_2}) = (\overline{x_1} * x_2)+(x_1 * \overline{x_2})+0+0 = (\overline{x_1} * x_2)+(x_1 * \overline{x_2})+(x_1 * \overline{x_1})+(x_2 * \overline{x_2}) = \overline{x_1}(x_1 + x_2) + \overline{x_2}(x_1 + x_2) = (x_1 + x_2)(\overline{x_1} + \overline{x_2}) = (x_1 + x_2)(\overline{x_1 * x_2})$

Схемата на този ЕДС след направената минимизация също се опростява:



Ясно се вижда, че елементите в схемата са вече по-малко – един "не", два "и" и един елемент "или", т.е. устройството става по-евтино от гледна точка на използваните логически елементи.

ЗАДАЧИ ЗА САМОСТОЯТЕЛНА РАБОТА

1) Попълнете таблиците за истинност на следните Булеви изрази:

а) $ABC + \overline{ABC}$

б) $ABC + A\overline{BC} + \overline{ABC}$

в) $A\,(B\overline{C} + \overline{B}\,C)$

г) $(A + B)(A + C)(\overline{A} + \overline{B})$

2) Опростете следните изрази, използвайки асоциативния логически закон:

а) $A \wedge \overline{B} + \overline{B} \wedge A + C \wedge D \wedge E + \overline{C} \wedge D \wedge E + E \wedge \overline{C} \wedge D$

б) $A \wedge B + A \wedge C + B \wedge A$

в) $(L \wedge M \wedge N)(A \wedge B)(C \wedge D \wedge E)(M \wedge N \wedge L)$

г) $F \wedge (K+R) + S \wedge V + W \wedge \overline{X} + V \wedge S + \overline{X} \wedge W + (R + K) \wedge F$

3) Използвайте формулите на де Морган за преобразуване на следните изрази:

а) $F = \overline{V + A + L}$

б) $F = \overline{A} + \overline{B} + \overline{C} + \overline{D}$

4) Опростете следните логически изрази:

а) $A = S \wedge T + V \wedge W + R \wedge S \wedge T$

б) $A = T \wedge U \wedge V + X \wedge Y + Y$

в) $A = F \wedge (E + F + G)$

г) $A = (P \wedge Q + R + S \wedge T)\ T \wedge S$

д) $A = \overline{\overline{D} \wedge \overline{D} \wedge E}$

е) $A = Y \wedge (W + X + \overline{\overline{Y} + \overline{Z}}) \wedge Z$

ж) $A = (B \wedge E + C + F) \wedge C$

5) Напишете логически израз за работата на NAND-елемент с четири входа.

6) Напишете ПНДФ за логическа функция на три променливи, дадена със следната таблица за истинност:

X_1	X_2	X_3	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

7) Напишете ПНКФ за логическа функция на три променливи, дадена с таблицата за истинност от задача 6).

8) Намерете минималната дизюнктивна форма на функцията от задача 6).

9) Намерете минималната конюнктивна форма на функцията от задача 6).

10) Направете синтез на логическо устройство по израза от пример 4 ж).

II. Технологични основи на елементната база и система елементи на изчислителните системи

1. Принципи на полупроводниковата биполярна технология.

При конструирането на цифрови устройства се използва стандартен набор от елементи, който се нарича **елементна база**. Към нея се предявяват редица изисквания като: ниска цена, голяма надеждност, ниска консумация на енергия, голямо бързодействие. Както данните, така и програмите се представят в компютъра на най-ниско хардуерно ниво като се използват стойностите на определени физически величини – ток, фазова разлика, напрежение и т. н.

В съвременните компютри за представяне на дадено число се използва двоичната бройна система. Двоичното число се изразява чрез двоичен сигнал, при който физическата величина на сигнала приема две ясно различими стойности, наречени **състояния**. Например, ако физическата величина, с която се представят сигналите, е напрежение, има две стойности за представянето на сигнала - 1 и 0:

При U ≤ а – се представя логическа 0-а , а при U ≥ в – логическа 1-ца.

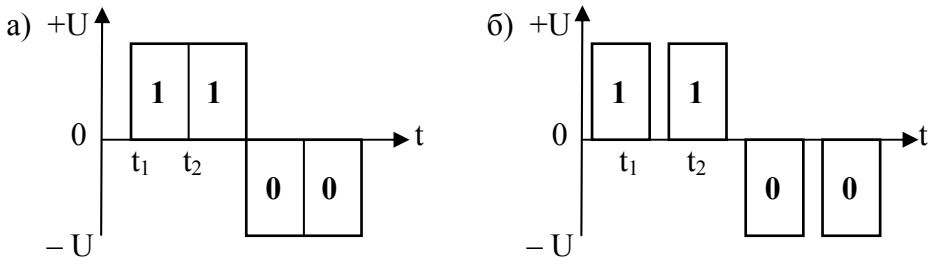
Местата, където реалният сигнал се изменя от едното състояние до другото, което не може да стане мигновено, се наричат *преден и заден фронт на сигнала*. Стойностите на предния и задния фронт на сигнала трябва да бъдат малки, т.е. сигналите трябва да имат стръмни фронтове, за да може реалният и идеалният сигнал да не се различават много. Използва се специален метод, при който сигналите се обработват и се създават стръмни склонове – това става посредством електрически схеми, наречени формировател. По принцип те могат да диференцират и интегрират сигнала, като променят стойностите на фронтовете му.

В зависимост от това с какви сигнали работят компютърните елементи, различаваме следните видове елементи:

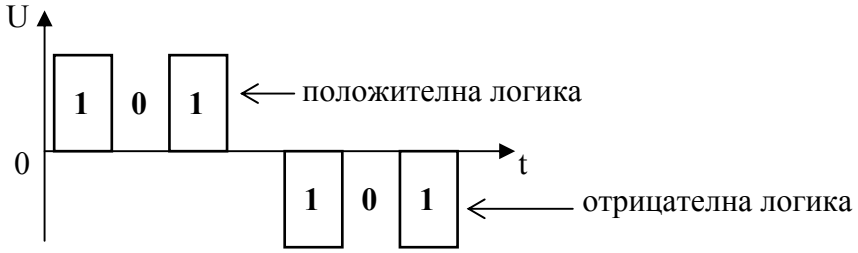
а) потенциални;

б) импулсни;

в) импулсно-потенциални.



При две нива на сигнала (заемащ само положителни стойности) липсата на сигнал се приема за 0-а, а наличието му за 1-ца.. Ако сигналът е с три нива, тогава има положителна стойност, която представя единица, и отрицателна стойност, която представя 0. Когато в работния елемент единиците се представят с положителната стойност на сигнала, а нулите нямат стойност, се казва, че този елемент е с положителна логика. Когато единиците се представят с отрицателна стойност, а нулите нямат стойност, елементът работи с отрицателна логика.



На тази база се изработват и електронните компютърни елементи, които съставят разгледаните вече логически елементи и се реализират с помощта на полупроводникови диоди и транзистори. Понеже последните консумират енергия, те се наричат *активни елементи*. Но в компютърните системи има и много други *пасивни елементи*, чиято функция е не да преобразуват и предават сигналите, използвайки енергия, а да я разпределят между активните елементи. Пасивни елементи са проводниците, резисторите, бобините и кондензаторите.

Елементите в компютърните системи по функции могат да се класифицират на *логически*, *формиращи* и *запомнящи*. Формиращите преобразуват формата на сигналите така, че да могат логическите елементи да работят нормално. Обикновено те са пасивни – резистори, бобини и др.

Запомнящите елементи са предимно активни, но могат да бъдат и комбинация от активи и пасивни. Те са предназначени да запомнят двоични цифри и обикновено са изградени с крайни автомати от типа спускови устройства – двупозиционни електронни елементи, способни да съхраняват едно от двете си състояния дотогава, докато външен сигнал не промени тези състояния.

Заедно изброените три вида елементи образуват т. нар. **система елементи на компютърната система**.

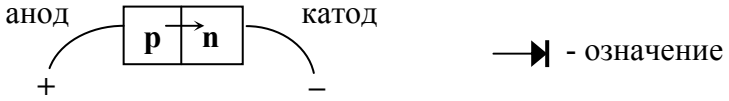
За реализация на активните елементи се използват полупроводникови технологии, основани на физиката на твърдото тяло (т. нар. p|n – преходи, тунелни и др. преходи в кристалите). Самите полупроводници са химически елементи, който в едни условия са проводници, а в други се проявяват като изолатори (диелектрици). Такъв например, селенът, който при облъчване със светлина става проводник, а на тъмно е изолатор. Причина за това е явлението фотоефект - под влияние на светлината в кристала на селена се избиват свободни електрони, които пренасят електрическия ток, и той става проводник.

Съществува и друга категория полупроводници, които предварително могат да бъдат подготвени така, че при определени електрически условия да станат проводници, а при други - изолатори (например Германий (Ge), Силиций (Si), Индий (In) и някои соли като Галиев Арсенид (GaAs) и др.).

Атомите в кристалната решетка на тези елементи са свързани по определен начин. Възможно е да липсват електрони на някои от атомите и като цяло те да бъдат положителни. Получените положителни заряди се наричат “дупки”. Възможно е също така да се получи и обратната ситуация – да има

излишък от електрони, които да провеждат електрически ток. И в двата случая има свободни токоносители и е възможно протичането на електрически ток – респективно от "дупки" или от електрони. Двата вида ток текат в противоположни посоки в полупроводника, когато той е включен към външен източник на електрическа енергия.

Според тези две възможности могат да се направят и два вида полупроводници: такива, които използват "дупки" – с дупчеста проводимост (**p**-полупроводник), и с електрони – с електронна проводимост (**n**-полупроводник). Когато двата вида се долепят много близо един за друг, на границата помежду им се получава зона, наречена *потенциална бариера*. При прилагане на външно електрическо поле, както вече се каза, между двата полупроводника може да протече ток.



Полученото устройство се нарича полупроводников диод. Ако обърнем полюсите на електрическото поле, токът, който протича през диода, спира. Този полупроводник проявява вентилно действие – пропуска ток само в едната посока. Ако по такъв диод се пусне променлив ток, към изхода му ще преминават само положителните полупериоди на променливия ток, т.е. след диода вече ще има не променлив, а прав ток. Точно по такъв начин се изправя променливия ток, който обаче е във вид на положителни пулсации с честота 50 Hz. Чрез специални методи тези пулсации могат да се изгладят. Първият полупроводников диод е създаден през 1948 г. и се е използвал за тази цел.

Тук обаче възниква въпросът може ли да се управлява големината на тока, протичащ през такова полупроводниково устройство? Това е възможно, ако се създаде видоизменено устройство, в което са долепени елементи в последователност **p-n-p**. Обикновено p-полупроводниците са по-големи, а n-полупроводникът е тънък. Такова устройство има три полюса – единият емитира ток (**емитер**), другият представлява изхода, където се събират токоносителите (**колектор**), а n-полупроводникът (**база**) управлява (пропуска или не пропуска) тока от емитера според това какъв е електрическият му потенциал.

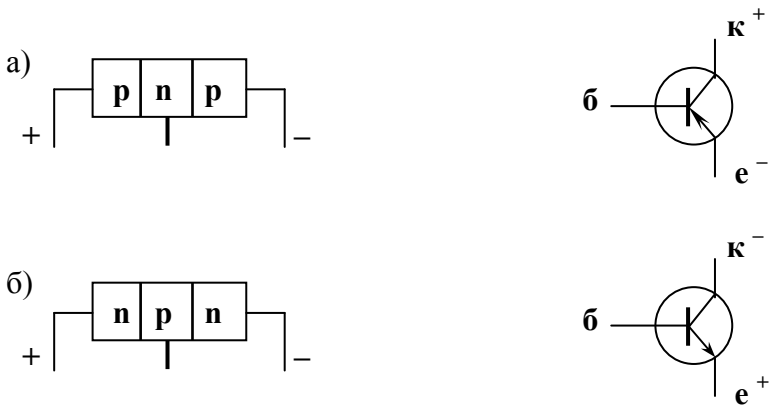
За да работи такова устройство, което се нарича **транзистор**, са необходими постоянни напрежения, включени към трите извода, които определят като цяло режима на транзистора. Ако към базата се включи променлив ток, в отрицателните полупериоди на тока тя ще пропуска тока от емитера към колектора, а в положителните – не, т.е. тя е управляващият електрод на транзистора.

В зависимост от постоянните напрежения на електродите му (изводите) транзисторът може да работи в два режима:

- ⇒ **ключов** – транзисторът е или отпушен (през него тече ток) или запушен (през него не тече ток) – т. е. устройството се проявява на практика като електронен ключ;
- ⇒ **усилвателен** – на базата се подава променлив ток и тя управлява плътността на тока през транзистора и на практика транзисторът усилва сигнала, като използва енергията на външния източник.

В компютърните системи основно се използва първия режим, но има случаи, където е възможно използването и на втория (при усилвателите на сигнали, например).

При транзисторите тип p-n-p емитерът е с отрицателен заряд, а колекторът с положителен. Те използва основно дупчеста проводимост (а)). Когато има подредба n-p-n, транзисторът е с електронна проводимост (б)), а зарядите на емитера и колектора са съответно “+” и “-“.



Германиевите транзистори са тип p-n-p, а силициевите n-p-n. Германият обаче, при температура 80^0 С се разтапя. Затова и първите транзистори, които били направени от този материал, били силно температурно зависими.

След германиевите в практиката навлизат силициевите транзистори, които са били основно използвани до 1960 г. Силицият има температура на топене 150^0 – 160^0 С. Технологията на изготвяне и на германиевите и на силициевите транзистори се нарича **биполярна**.

2. Принципи на MOS технологията.

Другият вид технология, която се използва за изготвянето на полупроводникови транзистори и диоди се нарича MOS (МОП – метал, окис, полупроводник).

Ако силицият се окисли, се получава силициев окис (SiO), който играе ролята на база. Върху него се поставя алуминиева пластина, изпълняваща ролята на колектор. Тогава се получава отново транзисторен компонент тип MOS, в който обаче трите електрода се наричат както следва: базата се нарича “врата” (gate), колекторът – “източник” (source), а емитерът - “отточен канал” (drain).

При този вид транзистори базата може да се направи много тънка, а колкото е по-тънка, тя има по-малка инертност и приборът работи по-бързо. Консумацията на енергия на целия прибор е много по-малка, което означава, че такъв елемент се загрява по-малко, плътността на компонентите му може да е по-голяма и технологията за неговото осъществяване е по-лесна.

Модерният вариант на MOS технологията се нарича **C-MOS III** (комплиметарен МОП III). При него се компенсират някои недостатъци – например по-ниската честота на работа (скорост) на първоначалните МОП транзистори. Първоначалните C-MOS елементи са били по-бавни от биполярните, поради някои химически особености на SiO. Сега този недостатък е изцяло компенсиран и вече има прибори, работещи с честота до 3-4 GHz!

Понастоящем всички компютърни елементи се правят по технологията C-MOS. Те са по-добри аналози на електроните лампи – управляват се с напрежение, а не с ток както биполярните полупроводникови елементи.

3. Технология на печатните платки.

За да се оформят по-големи звена и блокове (устройства) на компютърната система, и да се свързват отделните електронни схеми, както знаем, се използват **шини**. Това са група проводници, по които се предават електрически сигнали, представящи битове информация. Шините могат да бъдат еднопосочни и двупосочни, като подаването на битове става паралелно по всички проводници в дадена шина.

В практиката се използват два вида шини:

- такива, при които подаваните сигнали имат 2 състояния – за логическа единица потенциалът е положителен, а за логическа нула – нулев;
- шини, при които подаваните сигнали са с 3 състояния – висок потенциал за логическа единица, нулев за логическа нула и висок импеданс (елементът е включен, но нито приема, нито предава сигнали). За да се премине в третото състояние трябва да е налице специален разрешаващ сигнал.

Първите шини, а от там и блокове на звената и устройствата са били изработвани по обемната технология – отделните шини се правят с отделни изолирани проводници и със съответните запоявания към входовете и изходите на операционните устройства и елементи.

Днес се използва технологията на **печатните платки**. Те представляват изолационен материал (гетенакс, текстолит, стъклотекстолит), който от едната или двете страни има медно покритие (фолио). Върху него се разполага топологията на връзките между отделните операционни устройства и елементи (най-често се отпечатва или рисува с печатарско мастило). След това се правят отвори за входовете и изходите на елементите и получената рисунка се подлага на специална технологична обработка - потапя се в разтвор на азотна киселина или двужелезен трихлорид (Fe_2Cl_3) и това, което не е нарисувано, се разяжда, а останалото се запазва. Този процес се нарича “ецване” на платките. По този начин неразядените части оформят проводниците и връзките между операционните устройства и елементите.

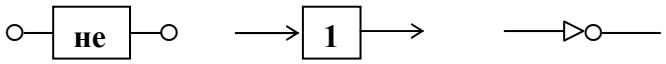
След това мастилото се измива най-често с разтвор на спиртна основа и в отворите за изводите върху платката се набождат елементите и устройствата. Изводите се запояват към металното фолио посредством специална машина - вана с разтопен калай и с метален вентилатор от ролков тип. Когато вентилаторът във ваната се върти, отгоре се образува калаена вълна (лента) и става спойка на изводите на елементите в платките. Накрая запоените изводи на елементите се покриват със специален защитен лак. Платката е готова да изпълнява ролята на модул на компютърната система.

4. Логически елементи.

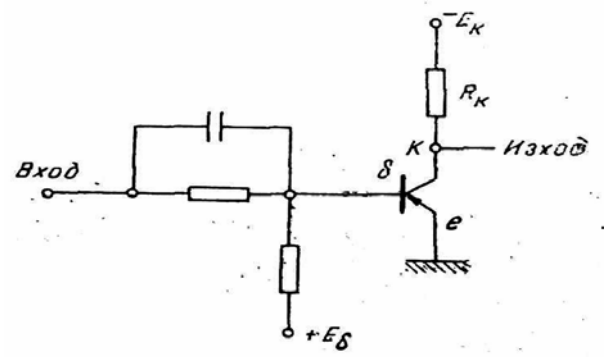
На базата на C-MOS полупроводниците се създават активните логически елементи. Чрез тях се конструират компютърните компоненти, изпълняващи логически функции. Всяка логическа функция се представя чрез дизюнкция (“или”), конюнкция (“и”) и отрицание (“не”), които физически се реализират посредством някаква елементна база или само чрез един елемент.

- реализация на логическия елемент “не” – означенията на елемента в литературата са посочени на фиг. 1. а), а електронната му схема представлява транзистор, режимът на който е подходящо избран и се нагажда със съответните постоянни напрежения (фиг. 1. б)):

а)



б)



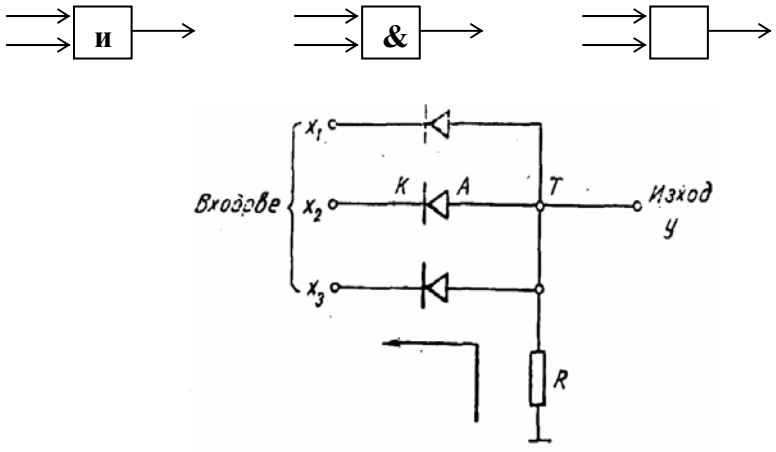
Фиг. 1. Схема на логическият елемент “не”.

Когато на входа на базата се пусне положителен импулс (логическа 1), транзисторът се отпушва, в резултат на което през него протича ток. При подходящо избрано съпротивление с ниска стойност този ток създава близко до нулата напрежение върху съпротивлението, което представлява логическа 0.

Схемата може да се реализира и с помощта на MOS-транзистор, като управлението се прави от напрежение, подадено към гейта на транзистора.

Тук не се отчита закъснението от превключването на елементите в схемата, но на практика има известно закъснение и изкривяване на сигнала. По принцип обаче, закъснението може да се пренебрегне в зависимост от големината на сигнала.

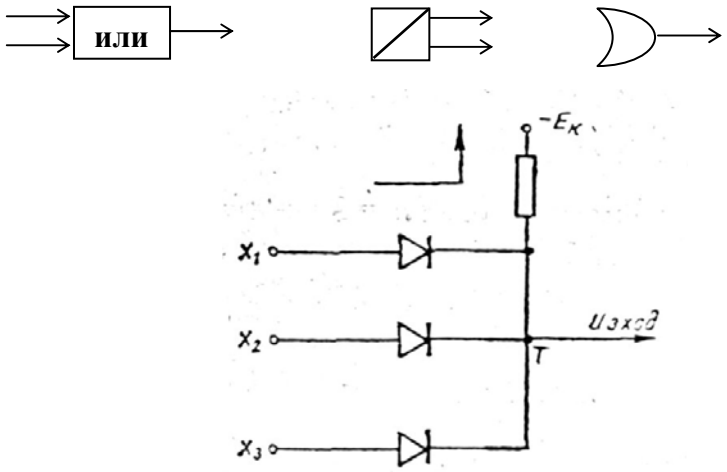
- <u>реализация на логическия елемент “и”</u> – този елемент се реализира посредством няколко диода (до 7), свързани в схемата показана на фиг. 2.:



Фиг. 2. Схема на логическият елемент “и”.

Ако на всички входове сме подали висок потенциал (логически единици), диодите са запушени и на изхода има 1. Ако на един от входовете на диодите е подадена логическа 0, ток протича в указаната на схемата посока и напрежението върху подходящо избраното съпротивление на изхода е ниско - това представлява логическа 0.

- реализация на логическия елемент “или” – реализацията му също е с диоди, но те са обърнати по отношение на тези в елемента “и”. Схемата му е посочена на фиг. 3.:



Фиг. 3. Схема на логическият елемент “или”.

Ако поне на един от входовете се подаде висок потенциал (логическа единица), единият диод се отпушва и през него протича ток в указаната посока, който минава през подходящо избраното съпротивление и отлага върху него високо напрежение представящо логическа 1. Ако на всички входове има нисък потенциал, ток не тече и на изхода също има нисък потенциал, т.е. логическа 0.

По същия начин се изграждат електронните схеми, които да реализират “стрелката на Пирс”, т.е. схема “и-не” с диоди и транзисторен елемент (NAND) и “щриха на Шефер” - схема “не или” с диоди и транзисторен елемент (NOR).

Съвременните компютри са изградени предимно с NAND-ове.

До тук ставаше дума за изчислителните системи със схеми от дискретни елементи, които вече са остарели. След тяхното използване хронологически възниква **интегралната технология** – отначало с интегрални схеми с ниска и средна степен на интеграция на елементите, а по-късно и големи интегрални схеми. Започва да се говори за *система от елементи*.

Обикновено при интегралната технология върху общ полупроводников кристал чрез единен технологически процес се конструират много елементи с еднакви характеристики на сигналите, с които работят. От биполярните схеми (много от които вече също са остарели) най-често се прилагат интегрални елементи с транзисторно-транзисторна логика (TTL), диодно-транзисторна логика (DTL), резисторно-транзисторна логика (RTL) и емитерно-свързана логика (ECL). При големите интегрални схеми предимно се използва MOS технология поради някои конструктивни съображения – по-нисък разход на енергия, по-малко разсейвана мощност и по-голяма възможност за микро-миниатюризация.

Поради напредъка на техническия прогрес и технологиите за производство на интегрални схеми границите на определенията за елемент, звено, и дори устройство постепенно се размиват. Днес отдавна съществуват цели устройства реализирани като една единствена голяма интегрална схема – микропроцесори, памети и др.

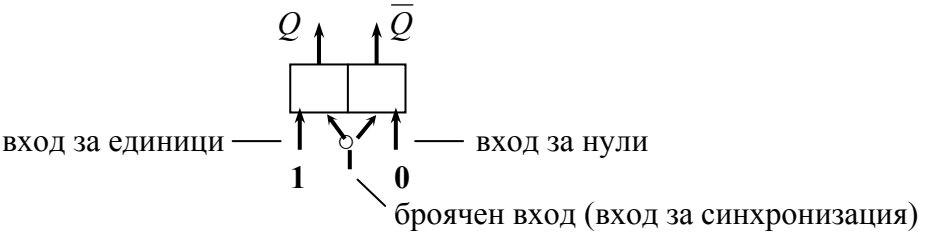
На практика технологията за подготвяне на кристала на полупроводника използва поредица от шаблони (маски), след което следва полагане на

различните материали (полупроводници, метали и др.) на слоеве през отворите на шаблоните. За целта се използват различни методи, много от които са химически или физически. Един от известните е този, използващ йонна имплантация.

5. Запомнящи елементи.

Двоичните запомнящи елементи имат предназначението да записват, съхраняват единици или нули, които се четат от тях чрез двоични сигнали. Те могат да са активни и пасивни. При активните елементи информацията е достъпна непрекъснато, докато пасивните не поддържат активен сигнал и при пристигането му трябва да бъдат активизирани.

Основните активни запомнящи елементи в съвременните компютърни системи са **тригерните схеми** (тригери, спускови устройства). Те са импулсно-потенциални елементи – на входовете им се пускат импулсни сигнали, а на изходите се изработват импулсни и потенциални сигнали. Когато се подаде импулс на входа, на изхода се получава определен потенциал. Смяната на потенциала е равнозначна на импулс. По същество тригерите принадлежат към класа на т. нар. крайни автомати. Всички тригерни устройства се отбелязват по начина, посочен по-долу - имат три входа и два изхода, т.е.:



Тригерът има две устойчиви състояния: когато в елемента е запомнена единица, състоянието е $Q = 1$ и $\overline{Q} = 0$, а когато е запомнена 0 – то е $Q = 0$ и $\overline{Q} = 1$.

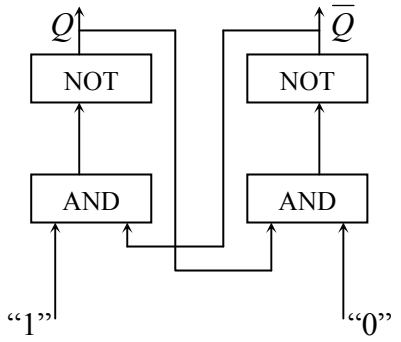
За преминаване (преобръщане) на тригера от едното в другото състояние могат да се използват и трите входа. Например, ако на входа за единици е подаден сигнал, на изхода Q излиза “1”. Ако се подаде сигнал на нулевия вход, тригерът преминава в състояние “0”. А когато се подават последователно сигнали на броячния вход, състоянието на тригера се променя алтернативно.

Тригерите се правят от логически елементи, като могат да се конструират различни конфигурации. Тригерите биват *несинхронизирани* и *синхронизирани* – при първите промяната на входните сигнали непосредствено въздейства на състоянието на изхода; при другите състоянието може да се промени само при наличието на импулс за синхронизация на броячния вход, който най-често се получава от тактовете на компютърния тактов генератор (clock, часовник).

Освен това според предназначението и сложността си тригерите могат да бъдат:

а) **RS-тригери** – могат да бъдат реализирани чрез логическите елементи NOT и AND (“стрелка на Пирс”) и работят с отрицателни импулсни входни сигнали. Както всички тригери и те запазват безкрайно дълго състоянието си, ако то не бъде променено от външен сигнал. При тях е забранено да се подават два сигнала “1” на двата входа, тъй като след тяхното преминаване не може да

се определи в какво устойчиво състояние ще се установи тригерът. Схемата на един несинхронизиран RS-тригер е посочена на фиг. 4.:

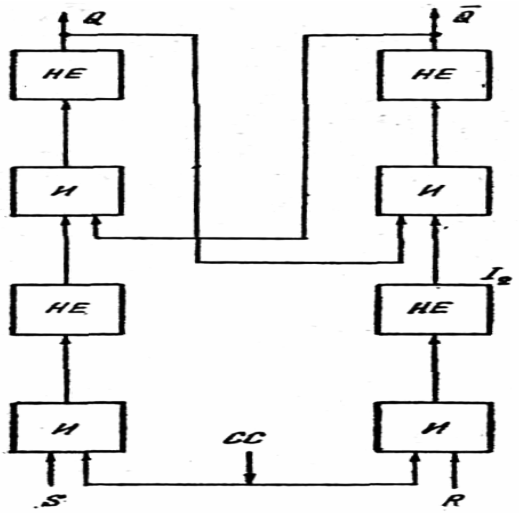


състояние "1" $Q = 1$ $\overline{Q} = 0$

състояние "0" $Q = 0$ $\overline{Q} = 1$

Фиг. 4. Схема на несинхронизиран RS-тригер.

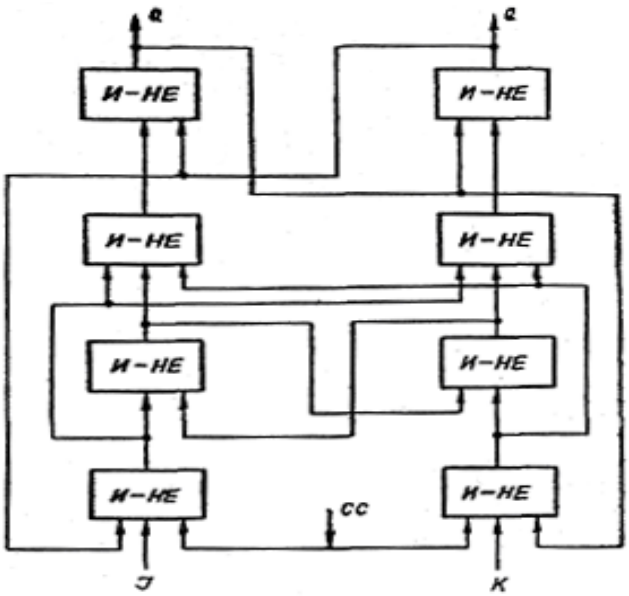
Схемата на синхронизиран RS-тригер е по-различна и изглежда по начина показан на фиг. 5. Вижда се, че този елемент съдържа входна логика и е с два входа: R (Reset) – нулиране, S (Set) – установяване:



Фиг. 5. Схема на синхронизиран RS-тригер.

Подадените едновременно сигнали на входове S и СС преминават като 1-ца през десния клон на тригера, която след елемента "не" става 0-а, влиза във входа на следващото "и" и предизвиква на изхода му 0-а, и накрая след инвертирането през последното "не" става 1, т.е. на Q-изхода се получава 1-ца и, която се записва в тригера. Проследяването на левия клон аналогично определя $\overline{Q}$-изхода да е 0-а. Това състояние на тригера може да се смени, ако на Reset и СС се подаде сигнал. Тогава той се обръща в състояние 0-а (т.е. Q=0, $\overline{Q} = 1$). Комбинацията R=S=1 е забранена по указаните по-горе причини.

б) **J-K тригери** – тяхната логическа схема е посочена на фиг. 6. Както се вижда този вид тригер е сложен, но много удобен, защото с незначителни изменения в схемата му могат да се получат всички останали видове тригери. Входовете му J и K съответстват на входовете R и S в RS- тригера. Тук обаче сигнали "1" могат да се появяват и на двата входа J и K, при което състоянието на тригера ще се променя алтернативно (т.е. получава се броячен вход).

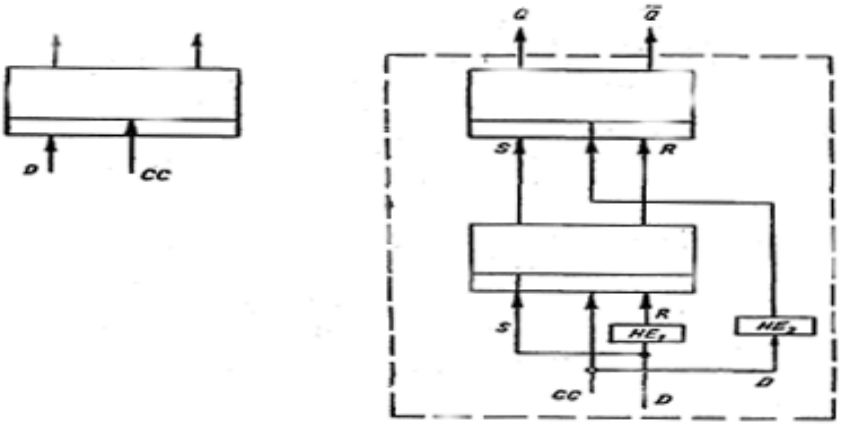


Фиг. 6. Логическа схема на J-K тригер.

Състоянията на този тригер при различни входни сигнали може да се видят на долната таблица и от нея лесно може да се проследи действието на схемата:

J (t)	K (t)	Q (t+1)
0	0	Q (t)
0	1	0
1	0	1
1	1	$\overline{Q}$ (t)

в) **D-тригери** – широко употребяван вид устройства, за които е характерно, че имат само един вход. Логическата им схема е показана на фиг. 7.:



Фиг. 7. Логическа схема на D-тригер.

В момент t под действието на синхронизиращия сигнал от СС, информацията постъпваща на D- входа се приема, но на изхода се появява един такт по-късно, т.е. Q (t+1) = D (t). Може да се види, че D-тригерът се състои от последователно свързани RS-тригери. При това презаписването на информацията от първия във втория се осъществява след записването ѝ в първия RS-тригер.

г) **NOR-тригери** – съдържат логическите елементи NOT и OR (“щрих на Шефер”) и работят с положителни сигнали. Тяхната схема е като на RS-тригерите, само че на мястото на елемента AND стои OR.

По принцип тригерите са основен елемент на операционните устройства, а също така и на съвременните компютърни памети.

6. Формиращи елементи и спомагателни компоненти.

Формиращите елементи играят ролята на оформящи сигналите схеми. Те променят не само силата, но и формата на сигналите (реактивните съпротивления, например). Такива са индуктивностите, капацитетите и комбинациите от тях, които променят фазата и формата на сигнала. Има и специални закъснителни елементи, които определят колко трябва да се забавят сигналите по пътя им от една към друга електронна схема.

Други спомагателни компоненти са т.нар. *тактови генератори* – те изработват и подават синхронизиращи сигнали към всички устройства и звена на компютърната система. Те представляват усилвателни схеми с положителна обратна връзка и се наричат още генератори на импулси. Генераторът трябва да изработва стабилни по честота импулси. В схемата на трептящия кръг на такъв генератор за стабилизация се включва кварцов кристал.

Накрая можем да споменем още, че всички полупроводникови елементи могат да бъдат не само в две, а в три състояния – отпушено, запушено и висок импеданс (съпротивлението е толкова голямо, че все едно веригата е прекъсната). Това свойство широко се използва в електронните схеми.

ВЪПРОСИ КЪМ РАЗДЕЛ II

1. Какви са предимствата и недостатъците на биполярната полупроводникова технология? За производството на какви полупроводникови прибори се използва тя?
2. С какво се различават биполярната и C-MOS технологии за производство на полупроводникови елементи?
3. След изследване на сайтове в Интернет, избройте последните варианти в C-MOS технологията. Посочете разликите между тях.
4. Какви материали се използват за производство на печатни платки? Посочете кои процеси в тази технология подобряват надеждността на връзките между елементите на електронните схеми.
5. Каква е разликата между полупроводниковите елементи, микро-модулите и интегралните схеми? Посочете колко поколения интегрални схеми познавате.
6. Каква е разликата между понятията “елемент” и “компонент”?
7. Какво е предназначението на логическите елементи и какви видове се използват в компютърните системи?
8. Опишете през какви етапи преминава еволюцията на запомнящите елементи на изчислителните системи.
9. Какво се разбира под понятието система елементи на компютърната система?

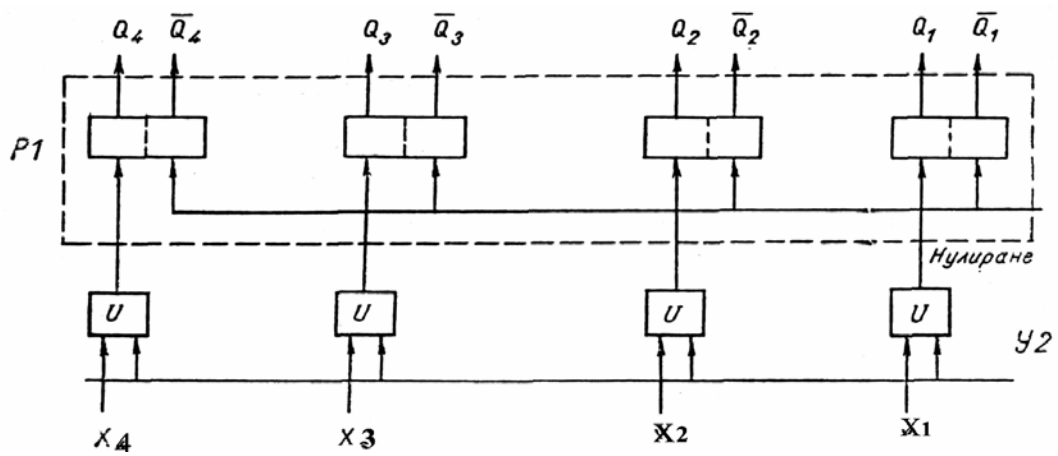
III. Възли и звена на изчислителните системи

1. Регистри.

Регистрите са устройства, предназначени за междинно запомняне на информация в процеса на нейното съхранение и за изпълнение на елементарни логически операции над нея. В тях се съхраняват двоичните кодове на думите за определено време. Те са съставени от запомнящи елементи, чийто брой зависи от разрядността на двоичните думи, запомнени в тях. Елементарните операции (микрооперации), които могат да се извършват в регистрите за един машинен такт, са следните:

- нулиране на регистъра – изчистване от цялата информация от него;
- приемане и предаване на двоична дума от един регистър в друг;
- преобразуване кода на двоичните числа (например от прав в допълнителен);
- поразрядно логическо събиране (дизюнкция) на две думи;
- поразрядно логическо умножение (конюнкция) на две думи;
- порязрядно логическо събиране на сума от модул 2 (логическа неравнозначност) на две думи;
- преместване разрядите на числата наляво или надясно.

Видовете регистри според начина на приемане, предаване и преобразуване на информацията са: *с паралелно действие* (приемат информацията от всички разряди на думата едновременно и могат да я предават паралелно), *с последователно действие* (отделните разряди на числото се приемат и предават последователно между елементите на регистъра за един машинен такт – преместващи регистри) и *с комбинирано действие*.



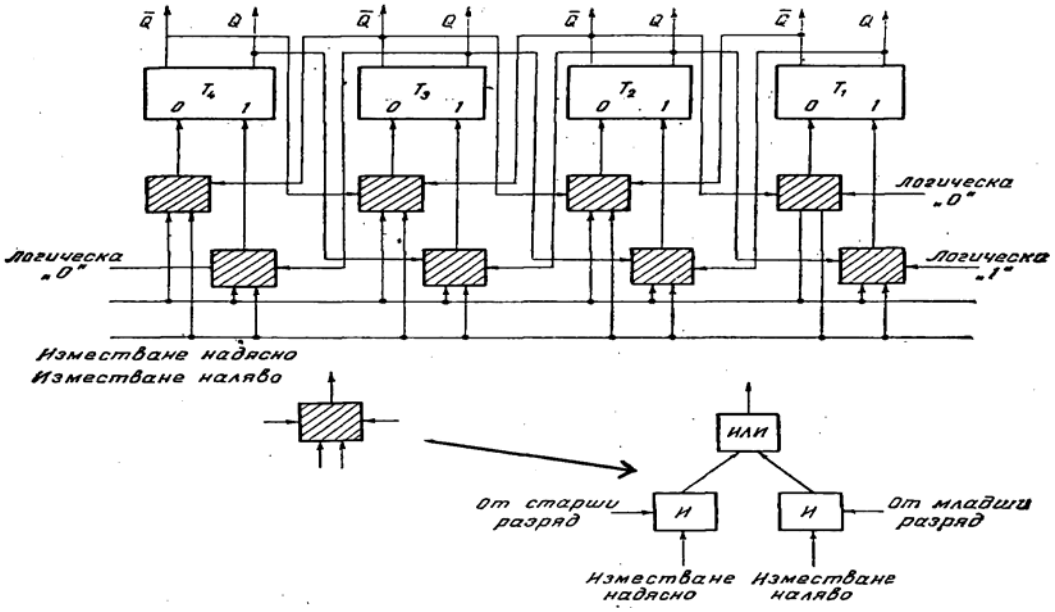
Фиг. 8. Схема на паралелен регистър.

Регистърът, показан на фиг. 8., работи на два такта – през първия се нулира съдържанието му чрез подаване на импулс по шината, свързваща всички нулеви входове на тригерите; през втория в него се приема новото съдържание едновременно по всички входове x. Когато старото съдържание е в регистъра и се подаде ново без да се извърши нулиране, се получава логическа операция над двете съдържания (думи) и новополученото съдържание (дума) ще бъде дизюнкция (логически сбор) от тях.

Микрооперацията за преобразуване кода на числата се реализира от изходите на тригерите, съставящи регистъра – ако от Q изходите на тригерите

се взема съдържанието на думата, записана в него, то от $\overline{Q}$ изходите се взема обратния код на числото (думата).

При реализацията на микрооперацията за преместване на разрядите на числата, за тяхното съхранение се използват т. нар. *преместващи регистри*, които могат да преместват числовите разряди надясно или наляво. При това онези разряди, които излизат извън разрядността на регистъра, се губят, а в освободените разряди се записват нули. Посоката на преместване се задава със специален управляващ сигнал. Схемата на преместващ регистър в двете посоки (реверсивен) е дадена на фиг. 9.:

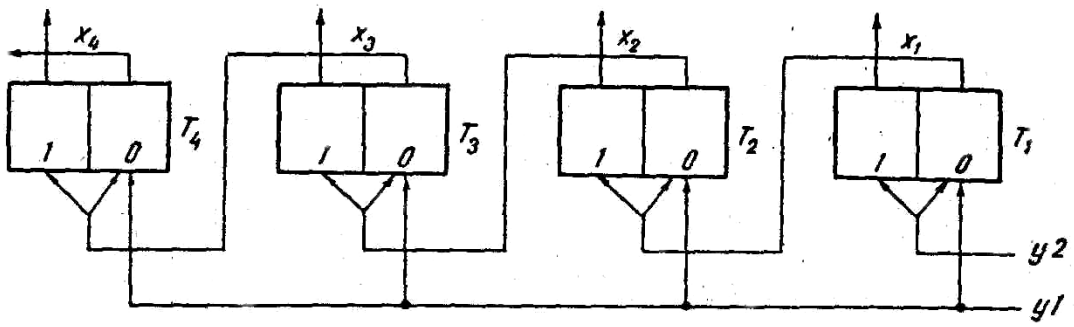


Фиг. 9. Логическа схема на преместващ регистър.

Операцията изместване наляво (към старшите разряди) се реализира по следния начин: предполага се, че в регистъра има някакво записано двоично число. Сигналът за изместване наляво се подава по управляващата шина към схемите "и". На другите им входове постъпват сигнали от изходите на младшия разряд. Ако той е в състояние 1 се изработва сигнал на изхода на схемата "и", който чрез "или" постъпва на единичния вход на дадения тригер. Ако той е в състояние нула, сигналът през втората схема "или" постъпва на нулевия вход. По такъв начин тригерът се установява в състояние, в което се е намирал младшият разряд, т.е. става преместване наляво. Аналогично се извършва преместването надясно, само че управляващия сигнал се подава по шината за изместване надясно.

2. Броячи.

Други основани звена в компютърните системи са броячите. Броячът е устройство, което преброява сигналите, постъпващи на входа му. Основните елементи, от които е съставен, също са тригери, но за разлика от регистрите тук се използва броячния вход (фиг. 10.). Броячът може да има от 0 до n-1 състояния, всяко от които запазва до пристигането на следващия входен сигнал. Ако той е предварително нулиран, всяко конкретно състояние изразява броя на импулсите, подадени и преброени от него.

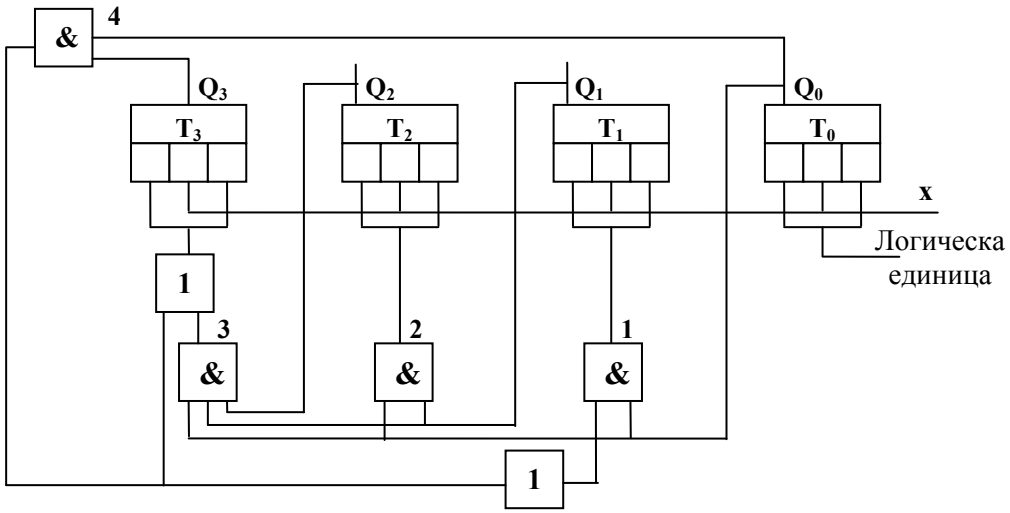


Фиг. 10. Схема на двоичен брояч с капацитет 4 разряда (16 състояния).

Връзката между съставните части на брояча (тригерите) е от съответния *Q*-изход на предишния тригер в схемата към броячния вход на следващия. Броячът работи с отрицателни импулси на броячните входове. Такива се образуват при спадане на потенциала на съответния *Q*-изход от младшия тригер.

Има броячи, които броят в намаление и се наричат *изваждащи*. Схемата им е същата, но сигналите за всеки следващ тригер се вземат от противоположните изходи. Най-често обаче броячите са комбинирани - прибавящи и изваждащи, и се наричат *реверсивни*. Посоката на броене при тях се определя от специален сигнал. Когато той е 1 – броячът брои в увеличаваща посока, когато е 0 – в намаляваща.

Възможно е да се направят и броячи, които да броят в друга бройна система. За целта се правят обратни връзки, за да се прескачат (съкратят) някои състояния. Примерната схема на брояч до 10 (десетичен брояч) с 4 тригера от този тип е показана на фиг. 11. До седмия импулс (0111) броячът брои нормално. Осмият импулс предизвиква обръщането на тригерите T_2 и T_1 и се получава 1110 (14) вместо 1000 (8). Следващият импулс преобръща само T_0 тригера и се получава 1111 (15). Десетият импулс нулира всички тригери и броячът започва отново да брои от 0, т.е. съкратени са състоянията от 9 до 13 и броячът брои до 10 (има 10 състояния – 0-7, 14, 15).



Фиг. 11. Схема на десетичен брояч с четири тригера.

Основните характеристики на броячите са:

- разрешаваща способност – минималното време, за което установява състоянието си след подаване на сигнал на входа (максимална честота на брояча);
- време за регистриране – времето, за което той установява състоянието си (скорост на работа);
- капацитет – максималния брой състояния, които може да регистрира (преброи).

3. Дешифратори (декодери).

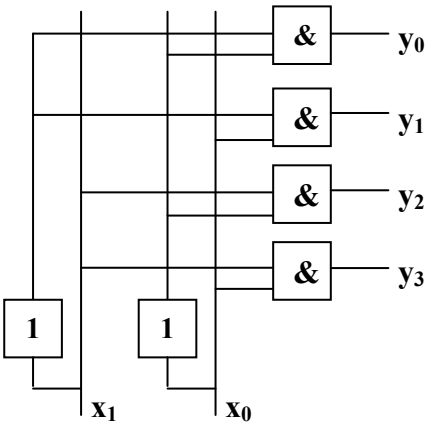
Дешифраторите са устройства, които по принцип имат n входа и 2^n изхода. Те преобразуват двоичния код на числата в управляващи сигнали - кодът постъпва на входа, а на един от изходите се получава изходният сигнал.

Дешифраторите реализират следната съвкупност от Булеви функции:

$$y_0 = \overline{x_{n-1}} * \overline{x_{n-2}} \dots \overline{x_1} * \overline{x_0},$$
$$y_1 = \overline{x_{n-1}} * \overline{x_{n-2}} \dots \overline{x_1} * x_0,$$
$$y_2 = \overline{x_{n-1}} * \overline{x_{n-2}} \dots x_1 * \overline{x_0},$$
$$\dots\dots\dots\dots\dots\dots\dots\dots$$
$$y_{2^n-1} = x_{n-1} * x_{n-2} \dots x_1 * x_0$$

В зависимост от изискванията за бързодействие и икономичност съществуват няколко типа дешифратори – линейни и многостъпални (правоъгълни и. пирамидални). Най-бързи са правоъгълните, а най-икономични по отношение разхода на апаратура са пирамидалните.

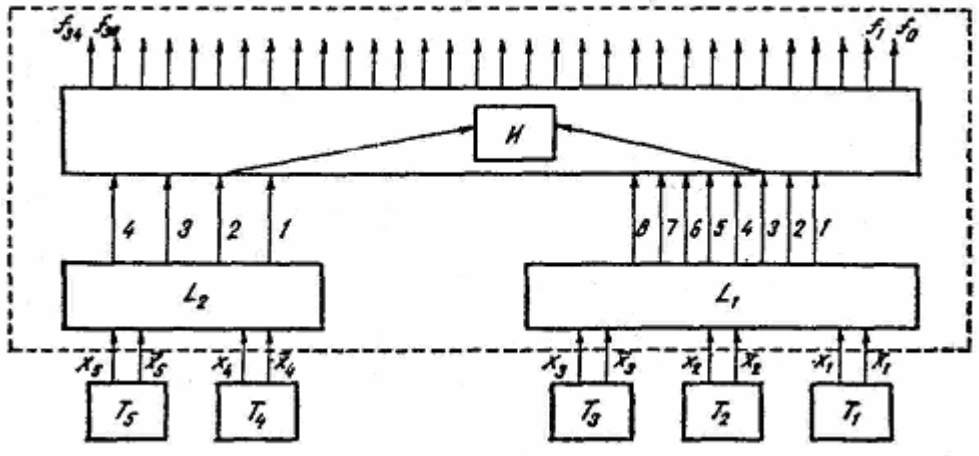
Схемата на линеен едностъпален декодер (връзката, която се реализира, изчерпва възможните състояния на изходите) е дадена на фиг. 12.:



Фиг. 12. Схема на линеен дешифратор.

При изграждането на дешифраторни схеми се използват тригери и логически елементи. При правоъгълния многостъпален дешифратор входната дума, която ще се дешифрира, се разделя на части и посредством линейни дешифратори се образуват изходни сигнали, които се наричат частични. Тази група линейни дешифратори образуват първото стъпало на правоъгълния

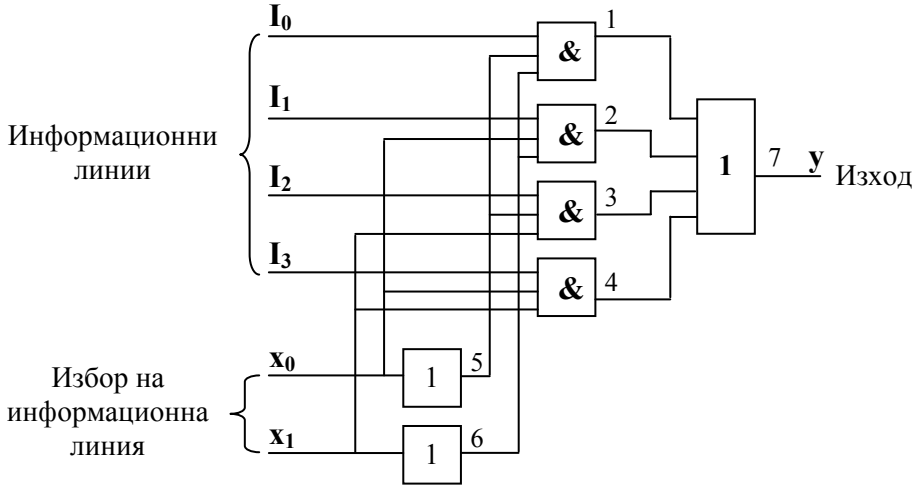
дешифратор (фиг. 13. – L1 и L2). На следващите нива се извършват конюнкции (операции "и") между различните двойки на частичните изходни сигнали, получени от предишните стъпала.



Фиг. 13. Схема на правоъгълен дву-стъпален декодер за 5-разредна дума.

В цифровите устройства често се налага да се подава информация от различни източници към обща магистрална линия (шина) – например, когато даден регистър трябва да се зареди с информация от няколко други регистъра. За целта се използват т. нар. **събирателна логика**. Тя се реализира физически чрез устройства, наречени *мултиплексори*, които са изградени от логически елементи. Пример за събиране на информация от 4 входа към **1** изход е показан на фиг. 14. Този мултиплексор реализира следното логическо уравнение:

$$Y = I_0 X_1 \overline{X}_0 \vee I_1 \overline{X}_1 X_0 \vee I_2 X_1 \overline{X}_0 \vee I_3 X_1 X_0$$



Фиг. 14. Схема на мултиплексор с четири входа и един изход.

Изборът на една от I-линиите става чрез линии x_0 и x_1: когато $x_1 = 0$ и $x_0 = 0$, се отпушва схемата 1 и към изхода y се пропуска линията I_0; когато $x_1 = 0$ и $x_0 = 1$, се пропуска I_1 и т.н.

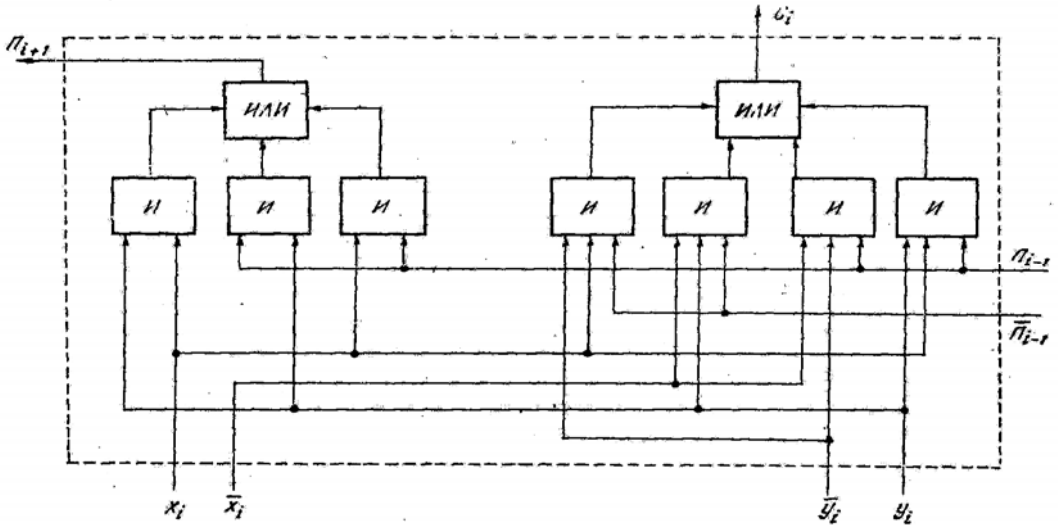
4. Суматори.

Основно звено в аритметичното и логическо устройство (АЛУ) на компютърния процесор е суматорът. Предназначението на суматорите е да събират двоични числа в съответните им прави и обратни кодове, но те могат да изпълняват и всички видове аритметични операции по определени алгоритми.

За да се определи стойността на даден резултат и преноса към разряда на двоичното число, е необходимо да се знаят цифрите на съответните числа и преноса от младшия разряд. Събирането се свежда до последователност от *n* на брой еднотипни операции, като се започва от най-младшия разряд. Всяка такава операция се осъществява от едноразряден двоичен суматор с три входа (ЕДС-3), по които се подават съответно разряди на двете числа *x* и *y* и преноса от по-младшия разряд $П_{i-1}$. Логиката на действие на ЕДС-3 се осъществява по следната система логически уравнения[4], а схемата, изградена съгласно тези уравнения, е показана на фиг. 15.:

$$C_i = \overline{П_{i-1}}\,\overline{X}_i\,Y_i \vee \overline{П_{i-1}}\,X_i\,\overline{Y}_i \vee П_{i-1}\,\overline{X}_i\,\overline{Y}_i \vee П_{i-1}X_i\,Y_i$$

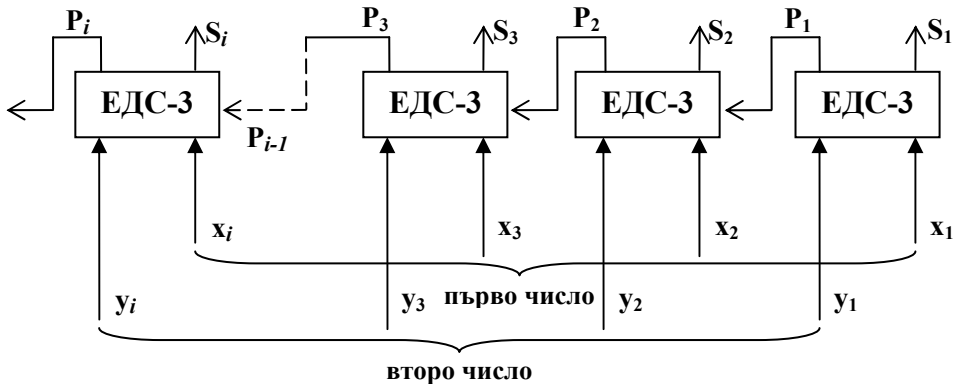
$$П_i = \overline{П_{i-1}}\,X_i\,Y_i \vee П_{i-1}\,\overline{X}_i\,Y_i \vee П_{i-1}X_i\,\overline{Y}_i \vee П_{i-1}X_i\,Y_i = X_i\,Y_i \vee П_{i-1}Y_i \vee П_{i-}$$



Фиг.15. Логическа схема на едноразряден двоичен суматор с три входа.

Когато се свържат преносите на няколко ЕДС се получава **многоразряден**. Този вид суматори се наричат **комбинационни**, защото нямат запомнящи елементи, т.е. двете събираеми не могат да се подават в различни машинни тактове, тъй като не се запомнят и затова разрядите на двете числа трябва да се подават едновременно. Освен това, щом се премахне сигнала на входа, сумата на изхода се загубва. Схемата на комбинационен многоразряден суматор е дадена на фиг. 16.:

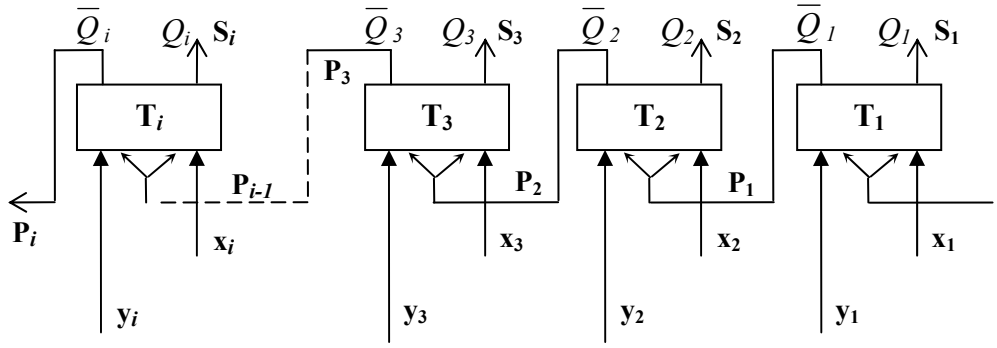
[4] Уравненията тук са дадени след минимизация на функциите за сума и пренос.



Фиг. 16. Схема на комбинационен многоразряден суматор.

Основна характеристика на комбинационните суматори е времето за образуване на сумата, т.е. времето от момента на подаване на входните сигнали до получаване на стойността. За да се получи правилна сума, всички преноси в i-тия разряд трябва да са преминали през всички разряди. Комбинационните суматори са бързи, но данните трябва да се подават едновременно на входовете им.

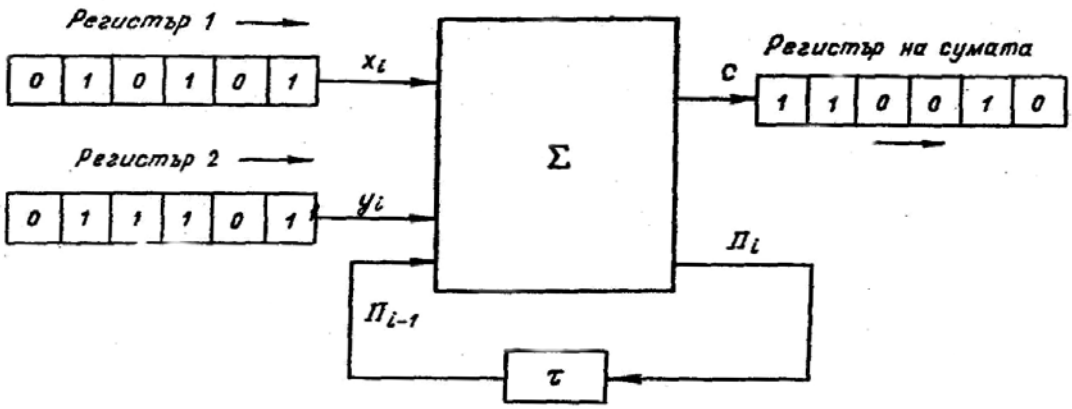
При натрупващите суматори като основни елементи се използват не ЕДС, а тригери. Тяхната логическа схема е показана на фиг. 17. по-долу:



Фиг. 17. Схема на многоразряден натрупващ двоичен суматор.

Ако възникне пренос, сигналът от изход $\overline{Q}$ се подава на входа на следващия тригер за коригиране, а събираемите се подават последователно, т.е. на два такта. Разрядността на суматора зависи от разрядността на машината.

Описаните два вида суматори са с паралелно действие, защото числата се подават паралелно на входовете им. Съществуват и суматори с последователно действие, при които разрядите на двете числа се подават един след друг на отделни машинни тактове. Физически такива многоразрядни суматори се реализират чрез едноразряден двоичен суматор и преместващи регистри. Схемата на такъв суматор е посочена на фиг. 18.:



Фиг. 18. Схема на многоразряден суматор с последователно действие.

ВЪПРОСИ КЪМ РАЗДЕЛ III

1. Посочете какво се разбира под понятието “микрооперация” в една компютърна система.
2. Какви микрооперации могат да се изпълняват в регистрите? Обяснете ги.
3. Какви видове регистри се използват в изчислителните системи и в кои техни устройства?
4. Какви видове биват броячите? Направете класификация на тези възли на изчислителните системи.
5. По какъв принцип се конструират броячи, които броят в бройни системи, различни от двоичната?
6. Какво е основното предназначение на декодиращите устройства? Къде в изчислителните системи намират приложение тези звена?
7. Посочете по какво се различават различните видове декодери.
8. По каква логика работят мултиплексорите? Напишете логическата им функция.
9. Какви аритметични действия могат да извършват суматорите? В какви устройства на компютърните системи намират приложение те?
10. Направете класификация на суматорите, използвани в компютърните системи. Посочете разликите между тях.

ПРИЛОЖЕНИЕ I

ВЪПРОСНИК
ПО ДИСЦИПЛИНАТА "КОМПЮТЪРНИ АРХИТЕКТУРИ"

1. Понятие за архитектура и организация на компютърните системи (КС). Обща структура и функции. Основни параметри на КС. Оценяване на възможностите на КС - скорост на процесорите, баланс на изпълнението и параметрите.
2. Еволюция на КС – поколения компютърни системи. Технологични нововъведения - полупроводникови памети, микропроцесори.
3. Класификация на компютърните системи. Съвременни КС от голям клас - организация и обща характеристика.
4. Съвременни мини- и микрокомпютърни системи – обща характеристика и особености. Еволюция на процесорите Pentium и Power PC.
5. Компютърни компоненти и функции на компютърната система - извличане и изпълнение на инструкции, възникване и обслужване на прекъсвания, управление на входно/изходните операции.
6. Структури за вътрешно свързване - свързване чрез шина. Структура на шината, управляващи линии. Многошинови йерархии, физическа специализация.
7. Вътрешно свързване на периферните компоненти. PCI шина - структура, команди, предаване на данни, арбитриране, множество прекъсвания.
8. Памети на компютърните системи - общ преглед, параметри, йерархия,
9. Полупроводникова основна памет - видове, организация, логика, пакетиране на чиповете, организация на модула на паметта, корекция на грешките - кодове.
10. Кеш-памет - принципи, елементи на организация, размер, съпоставяне (мапване) - видове.Директно и асоциативно съпоставяне.
11. Наборно-асоциативно съпоставяне - политика на записа, размер на реда, алгоритми на заместване, брой на кеш паметите.
12. Организация на кеш-паметта в процесорите Power PC и Pentium II, III и 4. Модерна DRAM организация на паметта - видове и характеристика.
13. Външна памет на компютърните системи. Магнитни дискове - организация, параметри.
14. Система RAID на магнитни дискове.
15. Оптична памет - CDROM, WORM, изтриваем оптичен диск, цифров видео-диск, магнито-оптични дискове. Външна памет на магнитни ленти.
16. Структура на централния процесор - организация на процесора и процесорните регистри. Цикъл на инструкцията в дълбочина - индиректен подцикъл, поток от данни.
17. Конвейер на инструкциите - стратегия на конвейера.
18. Организация и обработка на прекъсванията при процесорите Pentium и Power PC.
19. Допълнителни периферни устройства – скенери, цифрови камери и апарати, тунерни ТВ модули и звукови карти.
20. Допълнителни периферни устройства – принтери и плотери.
21. Архитектури с намален брой на системата инструкции (RISC). Резюме на компютърните иновации, използване на голям регистров файл (прозорци на

регистрите, глобални променливи) и оптимизация на регистрите на основата на компилатор.
22. Архитектура на RISC - особености и мотиви за разработката. Сравнение между CISC и RISC. Конвейерна система на RISC-процесорите.
23. Суперскаларни процесори – обща характеристика, подходи. Паралелизъм на ниво инструкции и машинен паралелизъм.
24. Суперскаларно изпълнение и използване в процесорите Pentium II и III.
25. Суперскаларно изпълнение и използване в процесора Pentium 4.
26. Суперскаларно изпълнение и използване в процесора Power PC.
27. Обща характеристика на процесорите с много голяма дължина на инструкцията (VLIW).
28. Микроархитектура на процесори с архитектура IA-64. Процесор Itanium.
29. Паралелна обработка на информацията. Видове системи за обработка на информацията – SISD, SIMD, MISD, MIMD.
30. MIMD компютърни системи – симетрични и клъстерни системи за паралелна обработка. Системи с нееднакъв достъп до паметта (NUMA).
31. SIMD системи за векторни изчисления.