

Noțiuni de teoria informației

Atunci când se face referire la capacitatea unui sistem de a transmite informații apare necesitatea de a exprima cantitatea respectivă de informații.

O informație este conținută într-un mesaj. Un mesaj reprezintă o succesiune de simboluri convenționale. Acestea pot fi :cuvinte, sunete, cifre etc.

Fie $S=\{s_1, s_2, \dots, s_n\}$ o mulțime de simboluri numite simboluri primare. Un bun exemplu de mulțime de simboluri primare o constituie alfabetul limbii române. Fiecare literă constituie în acest caz un simbol primar.

Fie $T=\{t_1, t_2, \dots, t_m\}$ o mulțime de simboluri numite simboluri secundare. Un simbol secundar reprezintă o combinație de simboluri primare. Un bun exemplu de mulțime de simboluri secundare este cel al mulțimii tuturor cuvintelor aparținând limbii române.

S-ar părea că numărul cuvintelor unei limbi ar putea constitui o măsură cantitativă a informației. Însă cuvintele nu au aceeași valoare în ceea ce privește precizia ce este adăugată ideii ce trebuie transmisă. În consecință, cuvântul nu poate servi ca unitate de măsură a informației. Printr-un raționament analog se concluzionează că nici litera nu poate constitui unitatea de măsură a informației. Însă putem considera că informația conținută într-un cuvânt trebuie să fie proporțională cu numărul de litere conținute de respectivul cuvânt. Mai general, cantitatea de informație conținută de un simbol secundar este proporțională cu numărul simbolurilor primare conținute.

Să considerăm totalitatea simbolurilor secundare ce corespund combinațiilor posibile de u simboluri primare. Numărul acestor simboluri secundare este n^u . Să considerăm de asemenea două sisteme pentru care n are valorile n_1 și n_2 . O aceeași cantitate de informație I reprezentată în cadrul fiecăruia dintre cele două sisteme se va scrie:

$$I = k_1 u_1 = k_2 u_2 \quad (1)$$

În acest cadru se impune următoarea egalitate:

$$n_1^{u_1} = n_2^{u_2} \quad (2)$$

Din (1) și (2) rezultă relația:

$$\frac{k_1}{\log n_1} = \frac{k_2}{\log n_2} = k_0 \quad (3)$$

Relația (3) va fi satisfăcută pentru toate valorile lui n numai dacă între k și n există relația:

$$k = k_0 \log n \quad (4)$$

Cum baza algoritmului este arbitrară, o vom alege astfel încât să rezulte:

$$k = \log n \quad (5)$$

Cantitatea de informație se poate acum exprima prin:

$$I = u \log n = \log n^u \quad (6)$$

Rezultă astfel o primă măsură practică informației: logaritmul numărului de combinații posibile a celor n simboluri primare pe lungimi de u simboluri.

Probabilitatea de apariție a unui simbol primar pe o poziție orecare în cuvântul de reprezentare a informației, este: $p = \frac{1}{n}$.

Relația (6) devine:

$$I = u \log \frac{1}{p} = -u \log p \quad (7)$$

Relația (7) constituie o expresie pentru cantitatea de informație. Este necesară în continuare stabilirea unei unități de măsură. Drept unitate de măsură a cantității de informație s-a luat măsura alegerii celei mai simple posibile, alegerea unei posibilități din două egal probabile. Deci, unitatea de măsură se definește pentru $u=1$, $n=2$ și $p=1/2$. Totodată logaritmul este considerat în baza 2. În concluzie, relația (7) conduce la:

$$I = 1 \cdot \log_2 2 = 1 \quad (8)$$

Această unitate informațională a căpătat denumirea de **bit** (de la *binary unit*) și a fost propusă în 1928 de către Hartley.

Se poate constata că frecvența simbolurilor primare în cadrul simbolurilor secundare nu este aceeași. Rezultă că nici probabilitățile de apariție a simbolurilor primare în cadrul simbolurilor secundare purtătoare de informație (în cadrul mesajului) nu sunt egale. Fiecărui simbol primar s_i i se poate atribui deci o probabilitate apriorică P_i . Ne așteptăm astfel ca apariția unui simbol primar ce prezintă o probabilitate mică să aducă mai multă informație. Astfel rezultă că dacă $p=1$ evenimentul este sigur și $\log p = \log 1 = 0$, deci informația conținută este nulă.

O sursă de informație emite o multitudine de mesaje diferite de lungime u . De aceea se convine să se exprime nu cantitatea de informație din fiecare mesaj ci *cantitatea medie* de informație pe un mesaj din setul respectiv de mesaje. Cum contribuția unui simbol s_i din mesaj este $-\log p_i$, *contribuția medie* a simbolurilor s_i din mesaj este $-u p_i \log p_i$ deoarece, pentru u suficient de mare, numărul simbolurilor s_i în mesaj se apropie de numărul lor mediu, $u p_i$, iar contribuția totală este numărul lor mediu înmulțit cu contribuția fiecăruia. Altfel spus, simbolul s_i poate să apară pe prima poziție a cuvântului sau pe cea de-a doua poziție sau pe cea de-a treia poziție sausau pe ultima poziție. Deci:

$$u p_i = \underbrace{\frac{1}{n} + \frac{1}{n} + \dots + \frac{1}{n}}_{\text{de } u \text{ ori}} = \frac{u}{n} = u \frac{1}{n} = u p_i$$

Aici s-a considerat cazul cel mai simplu, cel al probabilităților $p_i = \frac{1}{n}$ egale.

Informația medie totală conținută în mesajul format din u simboluri este suma contribuțiilor medii a tuturor celor n simboluri:

$$I_m = -up_1 \log p_1 - up_2 \log p_2 - \dots - up_i \log p_i - \dots - up_n \log p_n = -\sum_{i=1}^n p_i \log p_i \quad (9)$$

Informația medie pe simbol se va scrie:

$$H = \frac{I_m}{u} = -\sum_{i=1}^n p_i \log p_i \quad (10)$$

Mărimea H a căpătat numele de *entropie a sursei de informație*, deoarece formal, ca expresie matematică, ea este analogă entropiei termodinamice a unui sistem, stabilită de Boltzmann.

Observație: *Semnificația fizică a măsurii cantității de informație astfel obținută este aceea că ea dă numărul minim de operații (de alegeri) necesare pentru a ridica nedeterminarea.*

STRUCTURA DE BAZĂ A UNUI SISTEM NUMERIC DE CALCUL

Un sistem electronic de calcul, bazat în funcționarea sa pe principii numerice de codificare și procesare a informației, poate fi reprezentat prin intermediul schemei bloc din figura 1.

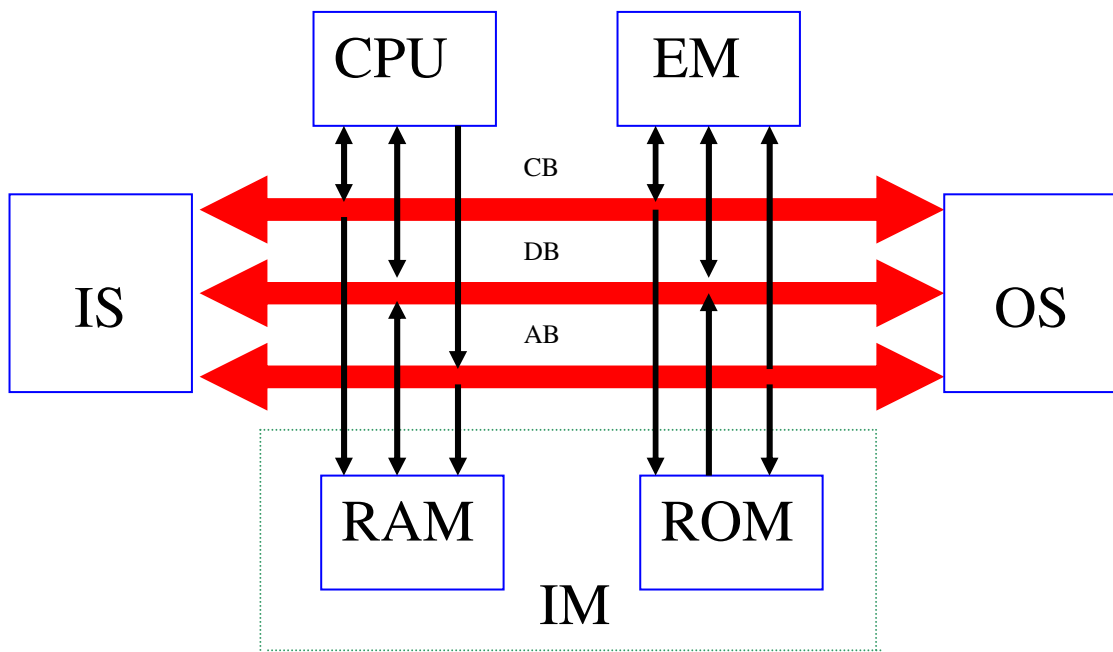


Figura 1

Blocul notat CPU (Control Process Unit) constituie unitatea de control a procesării informației. Din punct de vedere fizic, acest bloc este construit în jurul unui microprocesor. El este racordat la magistralele sistemului (trasee fizice de circuit imprimat, fire electrice, fibre optice etc.), notate AD (Address Bus), DB (Data Bus) și respectiv CB (Control Bus), prin intermediul cărora comunică (prin semnale electrice) cu alte subsisteme.

O magistrală se caracterizează prin *lățime* (numărul traseelor electrice pe care se pot transmite la un moment dat ansamblu de biți sub forma unor nivele ridicate sau coborâte de tensiune). Magistrala de date este cea care caracterizează cel mai adesea un microprocesor. Astfel, vorbind de un microprocesor pe 8 biți ne referim în fapt la lățimea magistralei lui de date.

Memoria unui sistem de calcul constituie un element esențial, atât din punct de vedere formal cât și practic. Distingem în figură două subsisteme de memorie.

Primul constă în două blocuri funcționale notate RAM (Random Access Memory) și respectiv ROM (Read Only Memory). Memoria RAM este denumită memorie în acces aleator deoarece cererile de citire sau înscriere în locații sunt făcute în mod imprevizibil, după necesitățile procedurilor software aflate în execuție. Din punct de vedere fizic, memoriile RAM sunt chip-uri (circuite integrate) plantate pe placa electrică pe care se găsește și microprocesorul, numită placă de bază (mother board). Memoriile RAM au în componența lor tranzistoare bipolare sau cu efect de câmp configurate în structuri electronice de bistabili. Din această cauză aceste memorii sunt

considerate volatile, adică starea circuitelor bistabile se pierde dacă tensiunea de alimentare dispare. La revenirea tensiunii de alimentare vechea stare nu se mai reface. Memoriile RAM prezintă însă un avantaj demn de remarcat și anume faptul că ele pot fi citite ori de câte ori se dorește (citirea este nedistructivă) și de asemenea pot fi înscrise cu ușurință de un număr nelimitat de ori. Alături de memoriile RAM construite cu circuite bistabile (denumite și SRAM – Static RAM) se mai utilizează și memoriile RAM ale căror locații sunt realizate sub forma unui tranzistor (în general de tip cu efect de câmp, caracterizat printr-o impedanță mare de intrare) ce are cuplat în circuitul lui de intrare un condensator (ce poate fi încărcat sau nu cu sarcină electrică). Astfel de memorii se numesc de tip DRAM (Dinamic RAM) deoarece pentru asigurarea funcționării corespunzătoare a lor este necesară reîmprospătarea periodică (la intervale de ordinul milisecundelor) a sarcinii electrice pe condensator. Acest proces este cunoscut sub numele de *refresh*. Memoriile DRAM sunt mai lente în acces decât cele SRAM, însă înalt integrabile în comparație cu cele din urmă.

Un sistem de calcul necesită incorporarea unui set de rutine software care să asigure pornirea. Aceste programe nu se pot memora în memoriile de tip RAM deoarece acestea îndeplinesc funcția de memorare numai după alimentarea lor, deci numai după pornirea sistemului. În mod logic, sistemul va trebui să dețină și un alt tip de memorie, nevolatilă. O astfel de memorie poartă denumirea de ROM. Memoria ROM este realizată fizic tot sub forma chip-urilor semiconductoare. Din punct de vedere constructiv, o locație de memorie ROM poate fi realizată prin intermediul unui rezistor fuzibil. Dacă acesta este distrus voit în procesul de programare a chip-ului, calea electrică pe care o asigură nu mai este disponibilă și ca atare această stare poate fi citită din exterior. Dacă rezistorul nu a fost distrus, el continuă să asigure calea electrică corespunzătoare, starea respectivă putând fi de asemenea citită din exterior. Specific este însă faptul că o dată distrus, un rezistor fuzibil nu mai poate fi refăcut. Deci înscrierea memoriilor ROM poate fi făcută o singură dată, de regulă de către firma producătoare a sistemului de calcul. Deci memoriile ROM nu pot fi decât citite, nu și înscrise. Evident, acest lucru constituie un mare dezavantaj, astfel încât s-a încercat construirea altor tipuri de dispozitive nevolatile. Așa au apărut memoriile de tip REEPROM (REProgrammable ROM). Acestea se aseamănă ca structură funcțională cu memoriile DRAM. Diferențele constau în faptul că în cazul memoriilor REEPROM condensatorul aflat în circuitul de intrare al tranzistorului cu efect de câmp pierde sarcina stocată într-un timp mult mai lung (ani), putând memora astfel informații chiar și pe durata nealimentării sistemului. De remarcat că ștergerea vechii informații din REEPROM și înscrierea alteia noi nu se mai realizează cu ușurință și în orice caz nu sub sistem. Pentru acest lucru este necesară extragerea circuitului din sistem, ștergerea informației existente prin iradierea chip-ului cu radiație UV, plantarea lui în regim de laborator într-un dispozitiv special de programare (programatorul de EPROM – Erasable ROM) și înscrierea cu noua informație. În final, circuitul este reimplantat în sistem și din acel moment el va putea fi doar citit. Această operație de ștergere și rescriere a unui chip poate fi realizată doar de câteva ori, după care circuitul este compromis. Amintim aici și faptul că s-au pus la punct și circuite de memorie cu posibilități de ștergere electrică a vechii informații și înlocuirea acestora cu alta nouă, fapt care permite reinscrierea chip-ului chiar sub sistem. Chiar și în acest caz, însă, circuitul este considerat de tip ROM deoarece operațiile de înlocuire a informației nu sunt de regim curent. Memoriile de acest tip se numesc EEROM (Electrically Erasable ROM).

Revenind la figura 1 observăm că accesul la magistralele ale memoriilor RAM și ROM sunt diferite. Astfel, memoriile RAM se află în acces bidirecțional cu magistrala de date în timp ce memoriile ROM se găsesc în acces unidirecțional.

Un alt bloc important în sistem este cel notat EM (External Memory). Aici este vorba în fapt de suporturi de memorare externe și nu de chip-uri semiconductoare integrate. În această categorie intră discurile magnetice (floppy discurile și hard discurile) sau discurile optice.

Blocurile IS și OS notează sistemul intrărilor (Input System) respectiv sistemul ieșirilor (Output System). În cadrul sistemului intrărilor pot intra echipamente cum ar fi: tastatură, mouse, light-pen, screen-touch, scanner etc. În cadrul sistemului ieșirilor intră: monitorul, imprimanta, plotterul etc.

REPREZENTAREA MĂRIMILOR ÎN CALCULATOARELE ELECTRONICE

Calculatoarele electronice (computere, ordinatoare) fac parte din categoria *mașinilor informaționale*, adică ele prelucrează informație. Este necesar să se facă distincție între noțiunile de *informație* și de *suport al informației*.

CE (Calculatoarele Electronice) pot fi clasificate din mai multe puncte de vedere. O clasificare fundamentală poate fi făcută relativ la modul în care sunt reprezentate în calculator mărimile cu care acesta lucrează. Astfel, în unele tipuri de calculatoare, mărimile prezintă variație continuă, în timp ce la altele acestea variază discret. În primul caz între mărimea x reprezentată și mărimea y_c care reprezintă pe x în interiorul mașinii, subzistă o relație de forma: $y_c = y_c(x)$, unde y_c este o funcție continuă de argument x . Astfel de calculatoare se numesc *analogice*.

A 2-a categorie de mașini intră în marea familie a *automatelor cu număr finit de stări*. În particular, în cazul calculatoarelor, aceste automate se numesc *digitale*, *cifrice* sau *numerice*.

Calculatoarele analogice utilizează ca semnale suport pentru mărimile interne tensiuni electrice sau curenți cu variații continue în timp. Aceleași tipuri de semnale electrice vor fi utilizate și de către calculatoarele numerice, doar că variațiile lor vor fi discontinue. Astfel de semnale vor fi numite impulsuri electrice. În cazul sistemelor care nu pot avea decât un număr finit de stări, mărimile reprezentate sunt în prealabil cuantizate. Situația se prezintă ca în figura 2c.

Calculatoarele digitale pot fi clasificate din diferite puncte de vedere. Un prim criteriu se referă la modul în care informația sosește în punctele în care ea este prelucrată. În general se lucrează cu semnale electrice puse în corespondență cu numere. Pentru aceasta sunt utilizate impulsurile standardizate. Prezența unui impuls poate fi reprezentată prin cifra 1 iar absența sa prin cifra 0 (evident că o convenție inversă va fi la fel de valabilă). Rezultă că informația din calculator poate fi reprezentată printr-un număr binar, format cu cifrele 1 și 0.

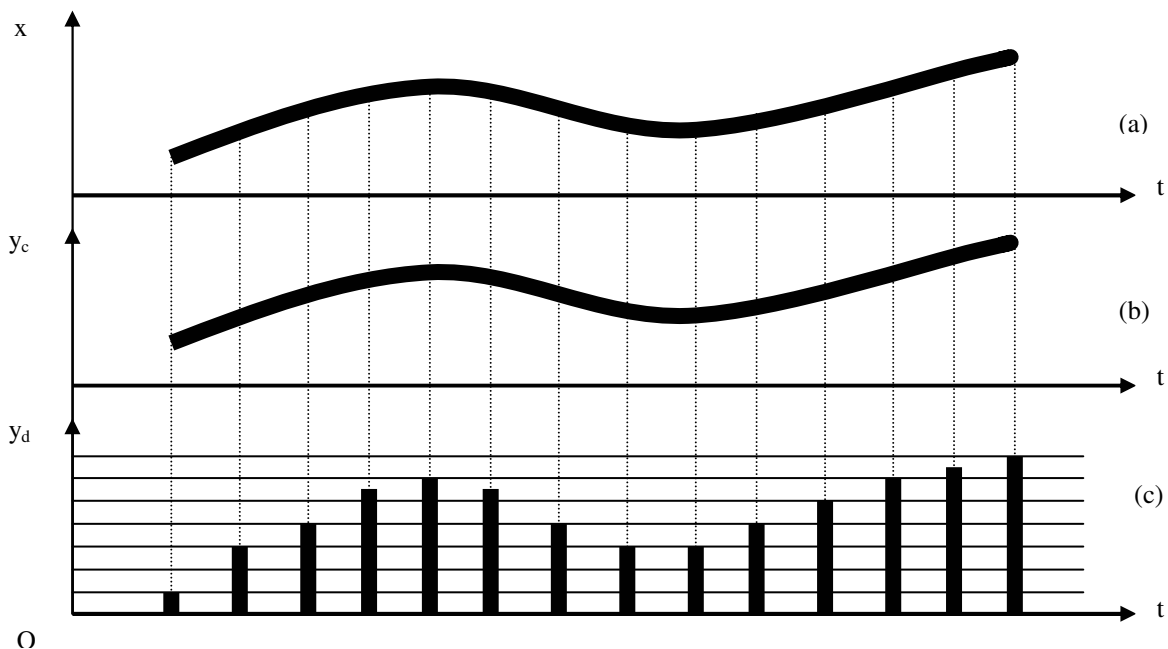


Figura 2

Calculatoarele la care toate instrucțiunile sunt transferate în paralel pe câte un canal se numesc *calculatoare de tip paralel*. Calculatoarele la care semnalele electrice, corespunzând diferitelor instrucțiuni, se transferă succesiv printr-un singur canal, se numesc *calculatoare de tip serie*. Evident, calculatoarele de tip paralel vor avea o viteză de lucru mare, în schimb vor fi deosebit de complexe, spre deosebire de calculatoarele de tip serie, care se vor caracteriza printr-o complexitate relativ scăzută dar timp de lucru mare.

Calculatoarele digitale pot fi clasificate și în funcție de natura parametrilor semnalelor electrice utilizate. Astfel, la unele calculatoare, în loc de a se utiliza impulsuri electrice, este folosită faza unei oscilații ca purtător al informației. În figura 3 sunt prezentate trei stări distincte care pot să apară într-un sistem de înaltă frecvență, în care există o inductanță neliniară.

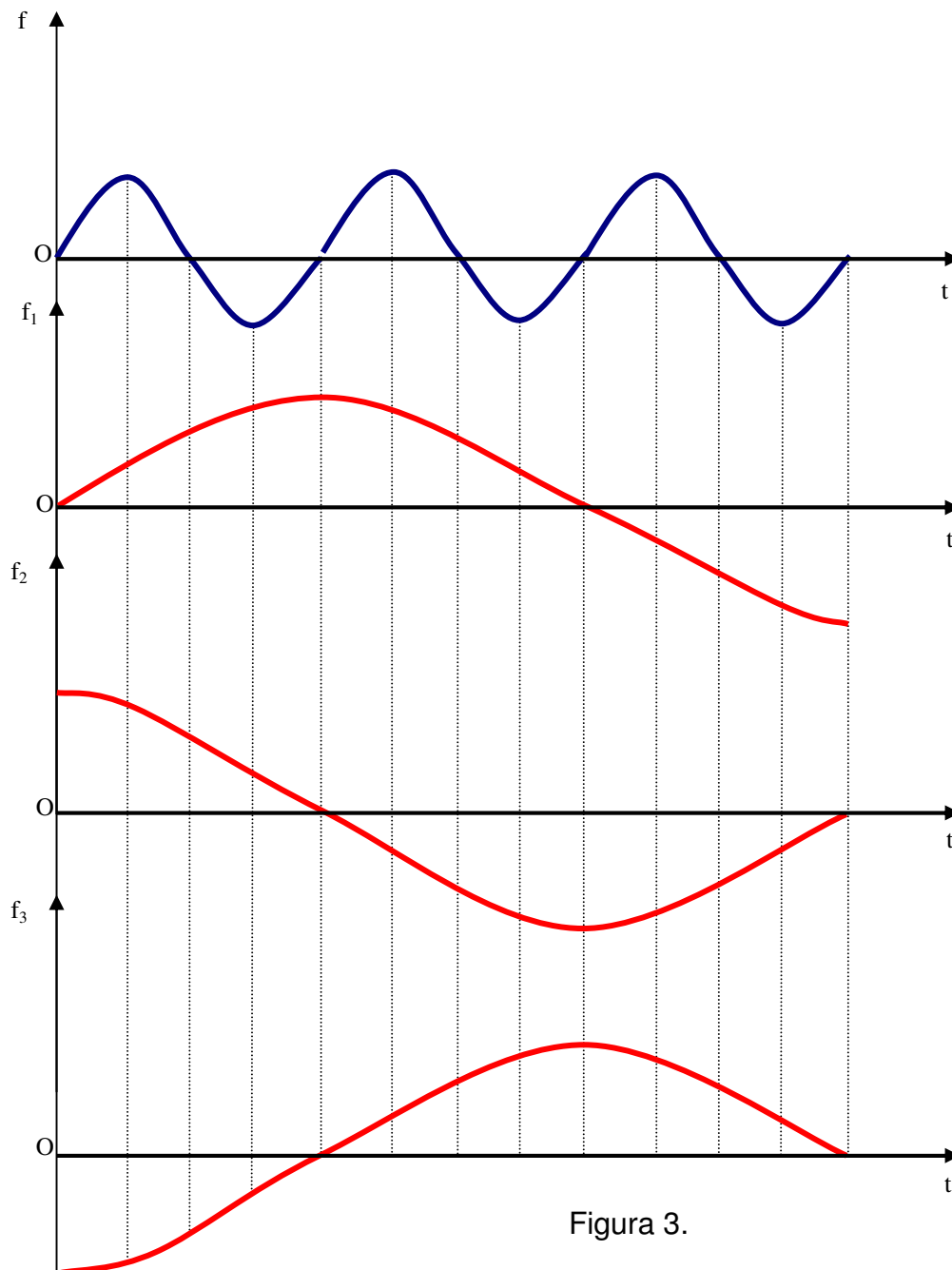


Figura 3.

Aceste stări pot fi puse în corespondență cu cifrele 0, 1, 2 (sau $-1, 0, 1$). Un astfel de mod de lucru este întâlnit în cazul calculatoarelor de tip *parametron*, numite astfel datorită faptului că oscilațiile subarmonice apar în sistemele parametrice, adică în sistemele în care parametrii sunt variabili în timp. La început erau utilizate ca elemente variabile inductanțele. S-au preferat apoi capacitățile variabile deoarece în felul acesta poate fi crescută viteza de lucru a sistemului. Se poate ajunge astfel la viteze de milioane de operații pe secundă.

Calculatoarele electronice digitale se mai pot clasifica după modul în care ele pot funcționa. Astfel, se remarcă calculatoarele cu *programare rigidă* și calculatoarele cu *programare elastică*. În ultima categorie intră *sistemele adaptive* sau *instruibile*, care pe baza experienței acumulate își pot modifica programul în funcționare. Această ultimă clasificare ține însă mai mult de structura software a sistemului decât de cea hardware.

SISTEME DE NUMERAȚIE

O clasificare a sistemelor de numerație cunoscute în momentul de față poate fi următoarea:

- sisteme de numerație poziționale;
- alte tipuri de sisteme de numerație.

Din prima categorie face parte sistemul de numerație în baza 10 sau cel în baza 2. Din cea de-a doua categorie poate face parte, de exemplu, sistemul de numerație roman.

Sistemul de numerație în baza 10 este cel utilizat în mod curent de operatorul uman în activitatea zilnică. În acest sistem fiecare cifră deține o pondere mai mare sau mai mică în reprezentarea valorii numerice la care participă, funcție de poziția pe care o ocupă în număr. Astfel numărul 1997 scris în baza 10, trebuie înțeles ca rezultat din formula:

$$1997 = 1 \cdot 10^3 + 9 \cdot 10^2 + 9 \cdot 10^1 + 7 \cdot 10^0$$

Aceeași cifră 9 are semnificații diferite, după poziția pe care o ocupă în număr.

Prin *bază* a unui sistem de numerație pozițional vom înțelege numărul de cifre al setului de simboluri utilizate de sistem. Sistemul de numerație zecimal are baza 10 deoarece utilizează 10 cifre (simboluri): 0, 1, 2, 3, 4, 5, 6, 7, 8 și 9.

De observat însă că orice număr natural mai mare decât 1 poate fi utilizat ca bază pentru un anumit sistem de numerație. Astfel, utilizând doar două simboluri (cifre), fie ele 0 și 1, vorbim despre sistemul de numerație *binar* ce utilizează *baza 2*.

În tabelul 1 sunt date, spre comparație, reprezentările primelor 17 numere în mai multe baze de numerație.

ZECIMAL	BINAR	TERNAR	OCTAL	DUODECIMAL	HEXAZECIMAL
0	0000	000	00	00	00
1	0001	001	01	01	01
2	0010	002	02	02	02
3	0011	010	03	03	03
4	0100	011	04	04	04
5	0101	012	05	05	05
6	0110	020	06	06	06
7	0111	021	07	07	07
8	1000	022	10	08	08
9	1001	100	11	09	09
10	1010	101	12	0m	0A
11	1011	102	13	0n	0B
12	1100	110	14	10	0C
13	1101	111	15	11	0D
14	1110	112	16	12	0E
15	1111	120	17	13	0F
16	10000	121	20	14	10

Tabelul 1

CODURI

Prin codificare se înțelege schimbarea formei de reprezentare a informației, ceea ce poate fi numită o traducere de limbaj.

Fie $X = \{x_0, x_1, \dots, x_{p-1}\}$ mulțimea simbolurilor primare emise de o sursă de informație, care urmează să fie codificate prin intermediul unor simboluri elementare aparținând unei mulțimi $Y = \{y_0, y_1, \dots, y_{n-1}\}$. Prin operația de codificare se asociază fiecărui element $x_i \in X$, $i = \overline{0, p-1}$, al sursei primare de informație, o secvență de simboluri $y_j \in Y$, $j = \overline{0, n-1}$. Corespondența astfel realizată va fi una biunivocă.

Notăm prin $Z = \{z_0, z_1, \dots, z_{p-1}\}$ mulțimea cuvintelor de cod. Astfel, cuvântul de cod z_3 poate fi exprimat, spre exemplu, ca:

$$z_3 = y_0 y_1 y_3 y_5$$

Putem defini în acest mod operația de codificare ca fiind o corespondență biunivocă între cele două mulțimi, X și Z , deci o aplicație bijectivă $f: X \rightarrow Z$. Codul se va numi uniform dacă toate cuvintele z_0, z_1, \dots, z_{p-1} au aceeași lungime. În tehnica digitală mulțimea Y este mulțimea binară $\{0, 1\}$, deci cuvintele mulțimii Z sunt cuvinte binare de o anumită lungime (în general 8 biți (octet sau byte), 16 biți, 24 biți, 32 biți, 64 biți etc.).

Simbolurile mulțimii X pot fi și altele decât cifre, deci pot rezulta mai multe tipuri de coduri. De remarcat sunt codurile numerice și cele alfanumerice. Exemple de coduri numerice sunt codurile: binar, octal, hexazecimal, zecimal codat binar (BCD- Binary Coded Decimal) etc. Exemple de coduri alfanumerice sunt: ASCII (American Standard Code for Information Interchange) și EBCDIC (Extended Binary Coded Decimal Interchange Code).

Codul ASCII codifică 128 de caractere (cele 52 de litere, majuscule și minuscule, ale alfabetului englez, cele 10 cifre zecimale, caractere speciale și caractere de comandă). Codul EBCDIC codifică 136 de caractere. Există caractere ASCII care nu au corespondent EBCDIC și invers.

Codul ASCII, datorită succesiunii caracterelor majuscule și minuscule, poate fi utilizat pentru ordonări alfabete.

Exemplu: Caracterul ? în ASCII are codul 3F iar în EBCDIC 6F. De asemenea, caracterele 0, A, a au codurile ASCII 30, 41, 61 iar cele EBCDIC F0, C1, 81.

Aplicație: Să se găsească codurile ASCII pentru caracterele: k, E, q, o, 7 știind că aceste coduri se succed liniar.

Rezolvare:

Caracter	ASCII
k	6B
E	45
q	71
o	6F
7	37

Codurile BCD presupun că mulțimea X este: $X = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ (mulțimea sursei primare de informație), iar mulțimea Z a cuvintelor de cod trebuie să conțină cel puțin 10 cuvinte distincte. Cuvintele de cod trebuie să reprezinte cel puțin 4 biți deoarece $2^3 < 10 < 2^4$. Se obțin în total A_{16}^{10} posibilități de codificare, Z fiind inclusă în mulțimea $\{0000, 0001, \dots, 1111\}$.

Din acest mare număr de coduri posibile, există anumite variante mai uzuale care pot fi divizate în două clase speciale de coduri: *coduri ponderate* și *coduri neponderate*.

Coduri ponderate

Fie $x \in \{0, 1, \dots, 9\}$ și $z_{(x)} = (y_3 y_2 y_1 y_0)_{(2)}$. Fiecărei cifre binare (bit) y_j , $j = \overline{0,3}$, i se atașează o pondere p_j astfel încât: $x = \sum_{k=0}^3 y_k p_k$.

Codurile ponderate cele mai utilizate sunt: 8421, 2421, 4221, 7421.

<u>Nr.</u>	<u>8421</u>	<u>2421</u>	<u>4221</u>	<u>7421</u>
0	0000	0000	0000	0000
1	0001	0001	0001	0001
2	0010	0010	0010	0010
3	0011	0011	0011	0011
4	0100	0100	0110	0100
5	0101	1011	1001	0101
6	0110	1100	1100	0110
7	0111	1101	1101	0111
8	1000	1110	1110	1001
9	1001	1111	1111	1010

Pentru codul ponderat 8421, în cuvântul de cod, bitul de rang 0 are ponderea 1, bitul de rang 1 are ponderea 2, bitul de rang 2 are ponderea 4 iar bitul de rang 3 are ponderea 8. Deoarece fiecare bit are ponderea numărului în binar și cuvintele de cod sunt chiar numerele succesive în sistemul binar natural, acest cod se mai numește codul zecimal binar natural (NBCD), în terminologie obișnuită fiind denumit (impropriu) codul BCD. Spre exemplu : $9=1 \cdot 8+0 \cdot 4+0 \cdot 2+1 \cdot 1$.

Aceași regulă de fixare a ponderii din cuvântul de cod, egală cu cea din notația codului, se respectă la toate celelalte coduri ponderate. Apar însă unele ambiguități în reprezentarea unor numere, ca de exemplu în cazul codului 2421. Aici, numărul 6 poate fi reprezentat atât prin codul 1100 cât și prin 0110. Într-adevăr, avem:

$$1 \cdot 2+1 \cdot 4+0 \cdot 2+0 \cdot 1=6 \text{ și } 0 \cdot 2+1 \cdot 4+1 \cdot 2+0 \cdot 1=6$$

S-a ales codul 1100 în virtutea respectării condiției de complementaritate:

$$C(6) = C(9) - C(3); \quad \overline{C(6)} = C(3).$$

Coduri neponderate

Exemple de coduri neponderate sunt:

- codul "exces 3"
- codul "binar reflectat"

Codul "exces 3" se obține din cuvântul de cod 8421 al cifrei zecimale respective, la care se adaugă 0011. Acest cod este util în situația sesizării informației inexistente pe suportul fizic (locație de memorie, registru etc. → se obține un "zero viu", codificat prin 0011).

Codul "binar reflectat" se obține prin "reflectări repetate" a codurilor pe $n-1$ ranguri, adăugând biți 0 într-unul din domenii și biți 1 în celălalt domeniu (cele două domenii sunt separate prin planul de oglindire).

Exemple de coduri binar reflectate: codul Gray și codul Gray închis.

Alte tipuri de coduri sunt codurile detectoare de erori și cele corectoare de erori.

Sistemul binar

În acest sistem sunt utilizate numai două cifre: 0 și 1 . Aceste cifre pot fi puse în corespondență cu absența respectiv prezența unui impuls electric. Marele avantaj al sistemului de numerație binar constă în numărul minim de cifre utilizate, ceea ce necesită un număr minim (2) de stări pentru elementele fizice care vor implementa structurile componente ale calculatorului.

Un alt avantaj îl constituie simplitatea algoritmilor de efectuare a operațiilor matematice. Dezavantajul ce poate fi amintit în acest caz constă în necesitatea lucrului cu șiruri lungi de simboluri în reprezentarea numerelor.

Conversia unui număr din reprezentarea în baza 10 în reprezentarea în baza 2 se realizează în următorii pași:

1. Se separă numărul zecimal în partea sa întreagă și partea fracționară.
2. Se convertește partea întreagă la o reprezentare în baza 2 .
3. Se convertește partea zecimală la o reprezentare în baza 2 .
4. Se combină cele două reprezentări (prin sumare), obținându-se reprezentarea binară a numărului zecimal dat.

Se remarcă în cadrul acestei descrieri existența a doi pași importanți (este vorba de pașii 2 și 3). Vom discuta pe larg acești pași:

a) Conversia unui întreg. Pentru a converti un număr scris în zecimal într-o reprezentare de tip binar, se împarte succesiv acest număr la 2 până când câtul devine 0 . Resturile obținute în urma acestor împărțiri succesive reprezintă cifrele numărului scris în noua bază. Aceste cifre (pe care le vom numi și biți) sunt calculate în ordine crescătoare, bitul cu rangul cel mai puțin semnificativ fiind primul.

Exemplul 1: Să se convertească numărul $254_{(10)}$ în binar.

$$\begin{array}{r} 254 \ | \ \underline{2} \\ 127 \ | \ 0 \\ 63 \ | \ 1 \\ 31 \ | \ 1 \\ 15 \ | \ 1 \\ 7 \ | \ 1 \\ 3 \ | \ 1 \\ 1 \ | \ 1 \\ 0 \ | \ 1 \end{array}$$

Rezultă: $254_{(10)} = 11111110_{(2)}$.

Exemplul 2: Să se convertească numărul $179_{(10)}$ în binar.

$$\begin{array}{r|l} 179 & \underline{2} \\ 89 & 1 \\ 44 & 1 \\ 22 & 10 \\ 11 & 10 \\ 5 & 11 \\ 2 & 11 \\ 1 & 10 \\ 0 & 11 \end{array}$$

Rezultă: $179_{(10)} = 10110011_{(2)}$.

b) Conversia unui număr fracționar. Pentru a realiza conversia unui număr zecimal fracționar în codul corespunzător binar, se va înmulți acesta cu 2 și se va separa apoi partea întreagă. Partea întreagă a produsului reprezintă un bit al numărului binar căutat. Procedura continuă până când partea fracționară devine nulă sau se obține precizia de reprezentare dorită.

Biții corespunzători reprezentării binare sunt determinați, prin acest procedeu, în ordine crescătoare, cel mai semnificativ fiind primul.

Exemplul 3: Să se convertească numărul $0.7109375_{(10)}$ în binar.

$$\begin{array}{l} 0.7109375 \cdot 2 \rightarrow 1 \\ 0.4218750 \cdot 2 \rightarrow 0 \\ 0.8437500 \cdot 2 \rightarrow 1 \\ 0.6875000 \cdot 2 \rightarrow 1 \\ 0.3750000 \cdot 2 \rightarrow 0 \\ 0.7500000 \cdot 2 \rightarrow 1 \\ 0.5000000 \cdot 2 \rightarrow 1 \end{array}$$

Rezultă: $0.7109375_{(10)} = 0.1011011_{(2)}$.

Evident că, acum, dacă vom dori conversia în binar a numărului zecimal $179.7109375_{(10)}$ acesta se va scrie:

$$179.7109375_{(10)} = 10110011.1011011_{(2)}$$

Ne punem în continuare problema conversiei inverse, din reprezentarea binară în cea zecimală. O posibilitate de a realiza acest lucru constă în scrierea sub formă de puteri a numărului reprezentat în binar.

Exemplul 4: Să se convertească numărul $10110011.1011011_{(2)}$ în zecimal.

$$\begin{aligned} 10110011.1011011_{(2)} &= 1 \cdot 2^7 + 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^1 + 1 \cdot 2^0 + 1 \cdot 2^{-1} + 1 \cdot 2^{-3} + 1 \cdot 2^{-4} + \\ &+ 1 \cdot 2^{-6} + 1 \cdot 2^{-7} = 128 + 32 + 16 + 2 + 1 + 1/2 + 1/8 + 1/16 + 1/64 + 1/128 = 179,7109375 \end{aligned}$$

O problemă de reprezentare binară este cea a numerelor zecimale negative. Există trei procedee de reprezentare: codul direct, codul complementar și codul invers. Un număr negativ din domeniul de lucru al calculatorului cu virgulă fixă este de forma:

$$a = -0.a_1 a_2 \dots a_n, \quad \text{unde } a_i \text{ are valoarea } 0 \text{ sau } 1.$$

În codul direct, convenim să exprimăm acest număr sub forma:

$$a_d = 1.a_1 a_2 \dots a_n = 1 + |a| = 1 - a$$

Rezultă că cifra zero se poate exprima în două moduri diferite:

$$\begin{aligned} +0_d &= 0.00\dots 0 \\ -0_d &= 1 - 0 = 1.00\dots 0 \end{aligned}$$

În primul caz se vorbește despre *nulul pozitiv*, iar în al doilea caz despre *nulul negativ*. De remarcat că utilizând codul direct, pot fi exprimate toate numerele cuprinse în intervalul $[-1(1-2^{-n}), 1-2^{-n}]$, evident cu n precizat.

În codul complementar, numerele negative se exprimă după regula:

$$a_c = 1.a'_1, a'_2, \dots, a'_n = 1 + a'$$

cu: $|a| + a' = 1$ sau $a' = 1 - |a|$

unde: $|a|$ este numărul pentru care a' reprezintă complementul față de 1.

Rezultă: $a_c = 1 + a' = 10 - |a|$

unde: a_c este numărul care pentru $|a|$ reprezintă complementul față de 2, iar 10 reprezintă codul binar pentru numărul 2 zecimal.

În codul complementar, pentru 0 avem o singură transformată. Cu ajutorul acestui cod se poate exprima orice număr din intervalul $[-1, 1-2^{-n}]$.

În codul invers, același număr negativ poate fi exprimat și cu ajutorul formulei:

$$a_i = 1.\bar{a}_1 \bar{a}_2 \dots \bar{a}_n$$

unde se utilizează regula:

$$\bar{a}_j = \begin{cases} 1, & a_j = 0 \\ 0, & a_j = 1 \end{cases}$$

Se observă că avem relația:

$$a_i + |a| = 1.11 \dots 1 = 10 - (10^{-n})$$

cu: a numărul care se obține prin complementarea fiecărui bit a_j , numerele fiind exprimate în codul binar, unde:

$$10^{-n} = \underbrace{0.00\dots 01}_{n \text{ cifre}}$$

Facem mențiunea anticipată că suma a doi biți poate fi:

$$\begin{aligned} 0 + 0 &= 0 \\ 0 + 1 &= 1 \\ 1 + 0 &= 1 \\ 1 + 1 &= 0 + \text{un bit } 1 \text{ de transfer} = 10 \end{aligned}$$

Revenind la relația de mai discutată mai sus, putem scrie:

$$a_I = 10 - (10^{-n}) + a$$

Cu ajutorul codului invers se obțin două expresii pentru 0 :

$$\begin{aligned} +0_i &= 0.00\dots 0 \\ -0_i &= 1.11\dots 1 \end{aligned}$$

Cu ajutorul codului invers se pot exprima numere din același interval ca și în cazul codului direct.

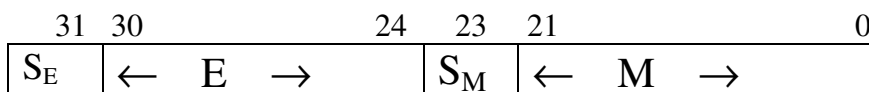
Forma normală de scriere a numerelor

În afara de reprezentarea în virgulă fixă a numerelor, este posibilă și reprezentarea acestora în virgulă mobilă. Într-o astfel de reprezentare numerele se scriu sub formă semilogaritmică.

Un număr reprezentat în virgulă flotantă prezintă două componente. Prima, pe care o notăm E , este denumită exponent și furnizează informații asupra ordinului de mărime al numărului. A doua componentă, notată M , poartă numele de mantisă și indică mărimea numărului într-un domeniu. Convenim ca $|M| < 1$. Atunci se poate scrie:

$$N = M * B^E$$

Ca exemplificare vom reprezenta un număr în virgulă flotantă pe patru octeți. S_E și S_M notează biții de semn ai exponentului, respectiv mantisei.



Dacă vom considera reprezentarea în cod complementar, exponentul va fi cuprins în intervalul $[-128, +127]$. Numărul cel mai mic ce poate fi reprezentat este:

$$a_m = \underbrace{(0.000\dots 01)}_{23 \text{ cifre}} \cdot 2^{-128} = 2^{-32} \cdot 2^{-128} = 2^{-151},$$

iar cel mai mare este:

$$a_M = \underbrace{(0.111\dots 11)}_{23 \text{ cifre}} \cdot 2^{+127} = (1 - 2^{-23}) \cdot 2^{+127} \cong 2^{127}$$

Operațiile aritmetice elementare în binar

Sistemul de numerație în baza 2 prezintă marele avantaj că permite imaginarea de algoritmi simpli pentru efectuarea operațiilor aritmetice. Adunarea se realizează utilizând următoarele reguli:

$$\begin{aligned} 0 + 0 &= 0 \\ 0 + 1 &= 1 \\ 1 + 0 &= 1 \\ 1 + 1 &= 10 \end{aligned}$$

Înmulțirea în binar se realizează astfel:

$$\begin{aligned} 0 \cdot 0 &= 0 \\ 0 \cdot 1 &= 0 \\ 1 \cdot 0 &= 0 \\ 1 \cdot 1 &= 1 \end{aligned}$$

Exemple: Adunarea în binar

$$A7.F + 29.4 = D1.3$$

$$\begin{array}{r} 10100111.1111 + \\ 00101001.0100 \\ \hline 11010001.0011 \end{array}$$

Înmulțirea în binar

$$39 \cdot 2A = 95A$$

$$\begin{array}{r} 111001 \cdot \\ \underline{101010} \\ 000000 \\ 111001 \\ 000000 \\ 111001 \\ 000000 \\ \underline{111001} \\ 100101011010 \end{array}$$

Se observă că înmulțirea se reduce la o succesiune de adunări.

Scăderea se realizează în mod analog adunării.

$$65.A - 15.E = 4F.C$$

$$\begin{array}{r} 1100101.101- \\ \underline{10101.111} \\ 1001111.110 \end{array}$$

În structurile numerice de calcul, scăderea se realizează sub forma unei sume algebrice între descăzut și scazătorul reprezentat în format de număr negativ. Împărțirea se reduce la o succesiune de scăderi. Un exemplu de împărțire între două numere reprezentate în binar, este următorul:

$$533 / B = 79$$

Trecută în binar, împărțirea arată astfel:

$$\begin{array}{r} 10100110011 \quad | \underline{1011} \\ \underline{1011} \quad \quad \quad \underline{1111001} \\ 10011 \\ \underline{1011} \\ 10001 \\ \underline{1011} \\ 01100 \\ \underline{1011} \\ 0001011 \\ \underline{1011} \\ 0000 \end{array}$$

Scăderea prin utilizarea complementului față de 2:

$$\begin{aligned}
 78 - 25 &= 78 + (-25) \\
 25_{(10)} &= 19_{(16)} = 00011001_{(2)} \\
 -25_{(10)} &= 11100111_{(2)} \Rightarrow 78 - 25 = 78 + (-25) \Rightarrow 4E_{(16)} - 19_{(16)} = 35_{(16)}
 \end{aligned}$$

Temă: Să se verifice toate exemplele prezentate trecând operațiile corespunzătoare în zecimal.

Observație: Adunarea a două numere în cod NBCD

Fie adunarea: $179 + 278$

$$\begin{aligned}
 179_{(10)} &= 0000\ 0001\ 0111\ 1001_{(8421)} \\
 278_{(10)} &= 0000\ 0010\ 0111\ 1000_{(8421)}
 \end{aligned}$$

Se începe adunarea cu cifrele mai puțin semnificative:

$$\begin{array}{r}
 9+ \\
 8 \quad \Rightarrow \\
 1 \leftarrow 7
 \end{array}
 \quad
 \begin{array}{r}
 1001+ \\
 1000 \\
 1 \leftarrow 0001
 \end{array}
 \quad
 \Rightarrow \text{rezultat greșit}$$

Deoarece procesorul adună în binar, cum adunarea a două tetrade implică o capacitate de reprezentare modulo 16, rezultatul poate fi greșit (în BCD capacitatea unei tetrade este modulo 10). Deci este necesară o ajustare a rezultatului prin adăugarea unei constante egale cu diferența dintre cele două baze, adică $10 - 16 = 6 \rightarrow 0110$, valoare numită și ajustare zecimală.

$$\begin{array}{r}
 \text{Avem astfel:} \\
 1001+ \\
 \underline{1000} \\
 1 \leftarrow 0001+ \text{ rezultat incorect în BCD} \\
 \underline{0110} \\
 1 \leftarrow 0111 \quad \text{rezultat corect în BCD}
 \end{array}$$

Ajustarea zecimală 6 (0110 în binar) trebuie făcută numai când suma celor două cifre BCD depășește valoarea 9, adică există un transfer în baza 10.

Algoritm:

- 1) Se efectuează suma $A+B$ și se testează dacă rezultatul $C \leq 9$.
- 2) Dacă da, acesta-i rezultatul. Dacă nu se trece la pasul următor.
- 3) Se efectuează suma $A + B + 6$ ($A + B + 0110$).

În cazul microprocesorului, adunarea a doi operanzi BCD se face în general cu o instrucțiune de adunare obișnuită urmată de o instrucțiune DAA (Decimal Adjust Accumulator).

ELEMENTE DE LOGICĂ MATEMATICĂ

Calculatoarele electronice digitale (numerice) efectuează operații logice. De aceea, pentru a putea înțelege principiile de operare ale subsistemelor de procesare logică, este necesar să facem o analiză a noțiunilor de logică matematică. Se disting mai multe direcții de preocupare în logica matematică, printre care logica claselor și logica propozițiilor.

În logica claselor se studiază relațiile dintre clasele (mulțimile) de obiecte, prin clasă înțelegându-se totalitatea obiectelor care au o anumită proprietate.

În logica propozițiilor se studiază propozițiile din punctul de vedere implicat de valorile de adevăr ale acestora, adică relativ adevărului sau falsității lor (este vorba de propoziții matematice).

De menționat că în afară de logica bivalentă, în care propozițiile pot fi fie numai adevărate, fie numai false, s-au dezvoltat și alte logici matematice în care se admit și alte valori pentru propoziții. Aceste logici au căpătat atributul de polivalente.

Majoritatea calculatoarelor electronice digitale lucrează în logică bivalentă, utilizând codificarea binară a informației. Există și sisteme care lucrează pe baza unor logici polivalente.

Fie A o propoziție. Dacă ea este adevărată vom scrie: $A = 1$. Dacă este falsă, vom scrie: $A = 0$. Astfel, 1 și/sau 0 reprezintă valori de adevăr (sau valori logice binare) pentru propoziția A . Expresiile în care intervin mai multe propoziții vor fi numite *funcții logice*.

Algebra logică binară a fost fundamentată pe lucrările matematicianului englez George Boole și din această cauză ea mai poartă și denumirea de *algebră Boole* sau *algebră booleană*. Ea are la bază o serie de postulate și teoreme.

Postulate

Fie M o mulțime oarecare de elemente, supusă la două legi de compoziție notate $+$ (sau \vee) și \cdot (sau \wedge), pe care le vom numi *sumă booleană* (sau operație de sumare booleană) și respectiv *produs boolean* (sau operație de multiplicare booleană), și o lege de complementare notată $-$ (sau \neg).

Elementele acestei mulțimi satisfac următoarele proprietăți:

1) Legea de idempotență:

$$x + x = x ; x \cdot x = x$$

2) Legea de comutativitate:

$$x + y = y + x ; x \cdot y = y \cdot x$$

3) Legea de asociativitate:

$$x + (y + z) = (x + y) + z = x + y + z$$

$$x \cdot (y \cdot z) = (x \cdot y) \cdot z = x \cdot y \cdot z$$

4) Legea de distributivitate:

$$\begin{aligned}x \cdot (y + z) &= x \cdot y + x \cdot z \\x + (y \cdot z) &= (x + y) \cdot (x + z)\end{aligned}$$

5) Există în M un element neutru pentru operația de sumare, notat „ 0 ”, astfel încât:

$$x + 0 = 0 + x = x$$

6) Există în M un element neutru pentru operația de multiplicare, notat „ 1 ”, astfel încât:

$$x \cdot 1 = 1 \cdot x = x$$

7) Legea de definire a complementarității:

Fie $x \in M$. Există $\bar{x} \in M$ (\bar{x} se numește complementul lui x) astfel încât:

$$\bar{\bar{x}} + x = 1 \quad \text{și} \quad x \cdot \bar{x} = 0$$

8) Legea de involuție:

$$\overline{\bar{x}} = x \quad (\text{se mai numește și legea dublei negări}).$$

9) Legile lui de Morgan:

$$\begin{aligned}\overline{x + y} &= \bar{x} \cdot \bar{y} \\ \overline{x \cdot y} &= \bar{x} + \bar{y}\end{aligned}$$

În general, notând cu \sum suma și respectiv cu \prod produsul boolean, relațiile De Morgan se scriu:

$$\overline{\sum_{k=1}^n x_k} = \prod_{k=1}^n \bar{x}_k \quad \text{și} \quad \overline{\prod_{k=1}^n x_k} = \sum_{k=1}^n \bar{x}_k$$

Teorema dualității

Dacă într-o expresie booleană **adevărată** înlocuiesc simbolurile $+$ și \cdot între ele, iar 0 cu 1 (și reciproc), se obține o altă expresie **adevărată**.

Exemplu:

$$\begin{aligned}x + \bar{x} &= 1 \\ x \cdot \bar{x} &= 0\end{aligned}$$

Observație: Operația (+) se mai numește SAU (OR) iar operația (·) ȘI (AND). Negarea (-) se mai numește NU (NOT).

Relații booleene utile:

$$1) \ x + 1 = 1$$

Demonstrație:

$$\underline{x + 1} = (x + 1) \cdot 1 = (x + 1)(x + \bar{x}) = x \cdot x + x \cdot \bar{x} + 1 \cdot x + 1 \cdot \bar{x} = x + 0 + x + \bar{x} = x + \bar{x} = \underline{1}$$

2) Proprietatea de absorbție:

$$\underline{x + x \cdot y} = x \cdot (1 + y) = \underline{x}$$

și relația duală: $\underline{x \cdot (x + y)} = x \cdot x + x \cdot y = x + xy = \underline{x}$

$$3) \ x + \bar{x} \cdot y = x + y$$

Demonstrație:

$$\begin{aligned} \underline{x + y} &= (x + y) \cdot (x + \bar{x}) \cdot (y + \bar{y}) = (x \cdot x + x \cdot \bar{x} + y \cdot x + y \cdot \bar{x})(y + \bar{y}) = (x + x \cdot y + \bar{x} \cdot y)(y + \bar{y}) = \\ &= x \cdot y + x \cdot \bar{y} + x \cdot y + x \cdot y \cdot \bar{y} + \bar{x} \cdot y + \bar{x} \cdot y \cdot \bar{y} = x \cdot y + x \cdot \bar{y} + \bar{x} \cdot y = x \cdot (y + \bar{y}) + \bar{x} \cdot y = \underline{x + \bar{x} \cdot y} \end{aligned}$$

$$4) \ x \cdot (\bar{x} + y) = x \cdot y \Leftrightarrow \text{duala celei precedente.}$$

Funcții booleene

Fie o variabilă booleană ale cărei valori depind de valorile altor variabile booleene. Vom spune că am stabilit o relație funcțională între variabilele respective.

Exemplu: Considerăm un grup de trei becuri, notate L_1, L_2, L_3 . Asociem fiecărui bec o variabilă booleană x_i care va căpăta valoarea 1 dacă becul L_i este aprins și 0 dacă becul este stins. Vom defini în continuare o mărime E numită *iluminare*. Vom spune că iluminarea este realizată ($E = 1$) dacă cel puțin două becuri sunt aprinse. Rezultă că E este o funcție de variabilele $x_i, i=1,3$, adică : $E = E(x_1, x_2, x_3)$.

Funcția E se va scrie:

$$E(x_1, x_2, x_3) = x_1 \cdot x_2 + x_2 \cdot x_3 + x_3 \cdot x_1 + x_1 \cdot x_2 \cdot x_3$$

Tabelul de adevăr al unei funcții logice

Valorile unei funcții logice de n variabile pot fi prezentate într-un tabel (numit *tabel de adevăr*) cu maximum 2^n linii și $n+1$ coloane. În primele n coloane se trec combinațiile de valori ale celor n variabile de intrare iar în ultima coloană se trec valorile corespunzătoare ale funcției. Vom conveni ca variabilele să fie considerate în tabel după indice descrescător, de la stânga la dreapta (x_n, x_{n-1}, \dots, x_1).

În cazul exemplului prezentat, tabelul de adevăr al funcției $E(x_1, x_2, x_3)$ este:

<i>Nr.crt.</i>	x_3	x_2	x_1	<i>E</i>
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

Vom nota valorile funcției prin: $E_k = F(x_1, x_2, x_3)$ unde $k_{(10)} = (x_1, x_2, x_3)_{(2)}$.
De exemplu: $E_5 = F(1, 0, 1)$, $5_{(10)} = 101_{(2)}$.

Operatori logici

Operatorii logici pot fi priviți ca funcții logice elementare definite pe domeniul X al variabilelor logice x_i .

Operatorii simpli sunt:

- negarea (complementarea) – operator unar;
- suma logică (disjuncția) – operator binar;
- produsul logic (conjuncția) – operator binar;
- echivalența – operator binar;
- suma modulo 2 – operator binar.

Operatorul de complementare.

Fie x o variabilă logică. Dacă asupra acesteia acționează operatorul de negare sau complementare, rezultatul va fi constituit de variabila \bar{x} , citită „non x ” sau „ x negat”. Simbolic putem scrie: $y = \neg x$, unde cu \neg s-a notat operatorul de complementare iar cu y rezultatul aplicării acestui operator asupra variabilei logice x .

O altă notație este următoarea: $y = \bar{x}$.

Tabela de adevăr corespunzătoare acestui operator este:

-	x	\bar{x}
	0	1
	1	0

Operatorul de complementare va mai fi numit și *NU (NOT)*.

Operatorul sumă logică.

Tabelul de adevăr este:

+	x_2	x_1	x_2+x_1
	0	0	0
	0	1	1
	1	0	1
	1	1	1

Un simbol acceptat este \vee - simbolul disjuncției logice. De aceea, operatorul acesta se mai numește și *disjuncție logică*. El mai este cunoscut și sub numele de *SAU (OR)* logic.

Operatorul produs logic.

Tabelul de adevăr este:

.	x_2	x_1	$x_2 \cdot x_1$
	0	0	0
	0	1	0
	1	0	0
	1	1	1

Se mai utilizează și simbolul \wedge - numit conjunție logică. Un alt nume pentru acest operator este cel de *ȘI (AND)* logic.

Echivalența logică.

Tabela de adevăr este:

\sim	x_2	x_1	$x_2 \sim x_1$
	0	0	1
	0	1	0
	1	0	0
	1	1	1

Un alt simbol acceptat este \Leftrightarrow . Echivalența logică mai este numită uneori și *coincidență* logică.

Suma modulo 2.

Tabela de adevăr este:

\oplus	x_2	x_1	$x_2 \oplus x_1$
	0	0	0
	0	1	1
	1	0	1
	1	1	0

Suma modulo 2 este echivalentă cu negarea operatorului de echivalență.

Implicația logică.

Tabela de adevăr este:

\subset	x_2	x_1	$x_1 \subset x_2$	$x_2 \subset x_1$
	0	0	1	1
	0	1	0	1
	1	0	1	0
	1	1	1	1

Noțiunea de implicație trebuie înțeleasă într-un sens special. Astfel, o propoziție falsă implică orice propoziție, iar o propoziție adevărată este implicată de orice propoziție.

Observații asupra operatorilor logici.

Dacă se consideră un calcul bivalent, atunci se pot imagina patru (2^2) operatori unari de tipul celui de complementare. Aceștia se pot reprezenta astfel:

α	x	αx	β	x	βx	γ	x	γx	δ	x	δx
	0	0		0	0		0	1		0	1
	1	0		1	1		1	0		1	1

Literele grecești α , β , γ și δ notează operatorii unari considerați. Cei patru operatori pot fi numiți:

- α - operatorul de anulare
- β - operatorul identitate (de repetare)
- γ - operatorul de negare (complementare)
- δ - operatorul de saturare

Dintre acești patru operatori nu prezintă interes decât cel de complementare, γ .

La fel, dacă se consideră toți operatorii care acționează asupra grupurilor de două variabile (operatori binari), rezultă $2^4 = 16$ astfel de operatori. Pentru operatorii unari am determinat 2^2 tipuri, deoarece formula este: $2^{(nr. liniilor din tabelul de adevăr)}$. Rezultă că pentru operatorii binari vom calcula 2^4 tipuri. Într-adevăr, pentru cele patru combinații posibile ale valorilor celor două variabile

de intrare, vom avea 2^4 posibilități de a preciza setul valorilor de ieșire. Nu toți acești operatori sunt esențiali în funcționarea structurilor numerice de calcul.

Disjuncția, conjuncția, echivalența și suma modulo 2 prezintă două proprietăți remarcabile: *comutativitatea și asociativitatea*. Putem exprima această observație prin intermediul următoarelor formule:

$$\begin{aligned} A \theta B &= B \theta A \\ (A \theta B) \theta C &= A \theta (B \theta C), \end{aligned}$$

unde θ este unul dintre operatorii enumerați (*oricare din aceștia pe una din pozițiile precizate*).

Forme normale.

Legile lui De Morgan, alături de următoarele relații:

$$\begin{aligned} x \sim y &= (x + \bar{y}) \cdot (\bar{x} + y) \\ x \oplus y &= (x \cdot \bar{y}) + (\bar{x} \cdot y) \end{aligned}$$

arată că este posibil să se reducă expresiile logice, în care intervin diverși operatori, la expresii în care nu intervin decât operatori de negare, disjuncție și conjuncție logică, numiți *operatori fundamentali*. Aceste forme se numesc *normale*. Se poate arăta că și relația de implicație se poate reda prin utilizarea unor operatori din setul celor trei fundamentali. Astfel expresiile: $\bar{x} + y$ și $\overline{x \cdot y}$ conduc la aceeași tabelă ca și $x \subset y$.

Formulele algebrei logice

Vom încerca în cele ce urmează o altă introducere în mod inductiv în algebra logică. Pentru aceasta vom admite următoarele:

1°. Literele latine mici (inclusiv literele cu indice), ca și constantele 0 și 1 sunt formule.

2°. a) Dacă expresia v este o formulă, atunci \bar{v} este de asemenea o formulă.

b) Dacă expresiile v și v' sunt formule, atunci expresiile:

$$\begin{aligned} v + v' \\ v * v' \\ v \supset v' \\ v \sim v' \end{aligned} \quad \text{sunt formule.}$$

3°. Nici o altă expresie în afara acestora nu este formulă.

Mărimile reprezentate prin litere mici latine se numesc și *variabile propoziționale*, în timp ce constantele 0 și 1 sunt constantele *fals* și *adevărat*.

Spre exemplificarea noțiunilor introduse mai sus vom arăta că expresia:

$$(((x \cdot y) \supset (x + y)) + z)$$

este o formulă.

Observăm că x , y și z sunt formule în conformitate cu punctul 1°. În conformitate cu punctul 2°, $x \cdot y$ și $x + y$ sunt de asemenea formule. Rezultă din 2°.b că:

$$((x \cdot y) \supset (x + y))$$

este de asemenea o formulă.

Astfel putem afirma că expresia de mai sus este cu siguranță o formulă.

Fiecare formulă din algebra logicii furnizează o funcție în această algebră. Se poate întâmpla ca două sau mai multe formule să conducă la aceeași funcție. Vom spune că două formule ϑ și ϑ' se numesc echivalente dacă furnizează aceeași funcție.

Ca exemplu se poate arăta că formulele: $(x_1 \cdot x_2) \cdot x_3$ și $x_1 \cdot (x_2 \cdot x_3)$ sunt echivalente.

Asemănător se poate verifica neechivalența formulelor: $\overline{x + y}$ și $\overline{x \cdot y}$.

Expresii analitice ale funcțiilor logice

(a) Conjunție și disjunție elementară

Vom numi conjunție elementară a unor variabile produsul logic al acestora sau al complementelor lor.

Exemplu:

$$Q_9^{(4)} = x_3 \overline{x_2} \overline{x_1} x_0$$

Convenim ca:

$$(1) \quad x^\sigma = \begin{cases} x, & \sigma=1 \\ \overline{x}, & \sigma=0 \end{cases}$$

Folosind convenția precedentă, conjunția de mai sus se scrie:

$$Q_9^{(4)} = x_3^1 x_2^0 x_1^0 x_0^1$$

Observând ultima expresie, putem atașa conjunției un echivalent binar (în cazul nostru $1001_{(2)} = 9_{(10)}$), specificat prin indicele inferior al notației Q , indicele superior reprezentând numărul variabilelor componente.

În general:

$$Q_k^{(n)} = x_{n-1}^{\sigma_{n-1}} \dots x_1^{\sigma_1} x_0^{\sigma_0} = \prod_{i=0}^{n-1} x_i^{\sigma_i}, \text{ cu } (\sigma_{n-1} \sigma_{n-2} \dots \sigma_1 \sigma_0) = K_{(10)}$$

Vom numi conjunții vecine două conjunții care sunt constituite din aceleași variabile și diferă doar prin complementarea unei singure variabile.

Exemplu:

$$Q_9^{(4)} = x_3 \bar{x}_2 \bar{x}_1 x_0 \quad \text{\textit{și}} \quad Q_1^{(4)} = \bar{x}_3 \bar{x}_2 \bar{x}_1 x_0$$

Observație: Prin sumarea logică a două conjuncții vecine se obține o conjuncție cu un număr de variabile mai mic cu o unitate, lipsind variabila ce diferă.

Exemplu:

$$Q_9^{(4)} + Q_1^{(4)} = \bar{x}_2 \bar{x}_1 x_0 (x_3 + \bar{x}_3) = \bar{x}_2 \bar{x}_1 x_0 = Q_1^{(3)}$$

Vom numi disjuncție elementară a unor variabile sumarea logică a acestora sau a componentelor lor.

Exemplu:

$$D_9^{(4)} = x_3 + \bar{x}_2 + \bar{x}_1 + x_0$$

Ținând cont de aceeași convenție ca în cadrul conjuncțiilor, vom scrie:

$$D_9^{(4)} = x_3^1 + x_2^0 + x_1^0 + x_0^1.$$

În general avem:

$$D_k^{(n)} = \sum_{i=0}^{n-1} x_i^{\sigma_i}, \text{ cu } (\sigma_{n-1} \sigma_{n-2} \dots \sigma_1 \sigma_0)_{(2)} = K_{(10)}$$

În mod similar cazului conjuncțiilor, se definesc disjuncțiile vecine:

Exemplu:

$$D_9^{(4)} = x_3 + \bar{x}_2 + \bar{x}_1 + x_0 \quad \text{\textit{și}} \quad D_1^{(4)} = \bar{x}_3 + \bar{x}_2 + \bar{x}_1 + x_0$$

Deci: $D_9^{(4)} = x_3 + D_1^{(3)} \quad \text{\textit{și}} \quad D_1^{(4)} = \bar{x}_3 + D_1^{(3)}$

Prin înmulțirea a două disjuncții vecine, dispare variabila ce se modifică în expresiile lor:

$$D_9^{(4)} \cdot D_1^{(4)} = (x_3 + D_1^{(3)}) \cdot (\bar{x}_3 + D_1^{(3)}) = D_1^{(3)}$$

b) Forma canonică disjunctivă a unei funcții (disjuncție de conjuncții sau mintermi).

Pentru o funcție $f(x)$ se poate scrie relația evidentă:

$$(2) \quad f(x) = x \cdot f(1) + \bar{x} \cdot f(0) \quad \left(\begin{array}{l} \text{dacă } x=0 \quad \text{avem } f(0) = 0 \cdot f(1) + 1 \cdot f(0) \\ \text{dacă } x=1 \quad \text{avem } f(1) = 1 \cdot f(1) + 0 \cdot f(0) \end{array} \right).$$

În mod similar, pentru o funcție de două variabile, avem relația:

$$f(x_1, x_0) = x_0 f(x_1, 1) + \bar{x}_0 f(x_1, 0) = x_0 [x_1 f(1, 1) + \bar{x}_1 f(0, 1)] + \bar{x}_0 [x_1 f(1, 0) + \bar{x}_1 f(0, 0)]$$

sau, utilizând convenția precedentă, avem:

$$(3) \quad f(x_1, x_0) = \sum x_1^{\sigma_1} x_0^{\sigma_0} f(\sigma_1, \sigma_0); \sigma_1, \sigma_0 \in \{0, 1\}.$$

În cazul general, se demonstrează prin inducție următoarea relație:

$$(4) \quad f(x_{n-1}, x_{n-2}, \dots, x_1, x_0) = \sum \left(\prod_{j=0}^{n-1} x_j^{\sigma_j} \right) \cdot f(\sigma_{n-1}, \sigma_{n-2}, \dots, \sigma_1, \sigma_0)$$

cu $\sigma_j \in \{0, 1\}$, $j = 0, 1, 2, \dots, n-1$.

În aplicații se poate utiliza o altă relație, și anume:

$$(5) \quad f(x_{n-1}, x_{n-2}, \dots, x_1, x_0) = \sum \left(\prod_{j=0}^{n-1} x_j^{\sigma_j} \right)$$

și $f(\sigma_{n-1}, \sigma_{n-2}, \dots, \sigma_1, \sigma_0) = 1$.

Deci expresia unei funcții logice în acest caz este formată din suma conjuncțiilor (mintermenilor) corespunzătoare valorilor 1 ale funcției.

Exemplu: Fie funcția descrisă prin următorul tabel de adevăr.

x_2	x_1	x_0	$f(x_2, x_1, x_0)$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

Deci, expresia lui f , conform relației precedente, este:

$$f(x_2, x_1, x_0) = \bar{x}_2 \bar{x}_1 x_0 + \bar{x}_2 x_1 x_0 + x_2 \bar{x}_1 \bar{x}_0.$$

c) Forma canonică conjunctivă a unei funcții logice (conjuncție de disjuncții sau maxtermi).
 Considerăm forma normal disjunctivă a funcției inverse, conform relației (4):

$$(6) \quad \overline{f(x_{n-1}, \dots, x_1, x_0)} = \sum \left(\prod_{j=0}^{n-1} x_j^{\sigma_j} \right)$$

și
$$\overline{f(\sigma_{n-1}, \dots, \sigma_1, \sigma_0)} = 1, \quad f(\sigma_{n-1}, \dots, \sigma_1, \sigma_0) = 0$$

Forma canonică conjunctivă a unei funcții logice se obține prin complementarea expresiei (6) și folosind relațiile De Morgan:

$$(7) \quad f = \overline{(\overline{f})} = \overline{\sum \left(\prod_{j=0}^{n-1} x_j^{\sigma_j} \right)} = \prod \left(\sum_{j=0}^{n-1} \overline{x_j^{\sigma_j}} \right)$$

și
$$f(\sigma_{n-1}, \dots, \sigma_1, \sigma_0) = 0.$$

Pentru exemplul precedent avem:

$$f(x_2, x_1, x_0) = (\overline{x_2} + \overline{x_1} + \overline{x_0}) \cdot (\overline{x_2} + x_1 + \overline{x_0}) \cdot (x_2 + \overline{x_1} + x_0) \cdot (x_2 + x_1 + \overline{x_0}) \cdot (x_2 + x_1 + x_0)$$

Funcții incomplet definite

Uneori, pentru anumite combinații ale valorilor variabilelor independente, nu este precizată valoarea funcției sau aceste combinații nu apar niciodată în sistemul fizic ce materializează funcția dată. Vom spune că aceasta prezintă valori indiferente. Valoarea indiferentă o vom nota cu X . În liniile în care funcția nu este definită s-a trecut valoarea indiferentă X .

Exemplu de funcție incomplet definită:

x_2	x_1	X_0	f
0	0	0	0
0	0	1	0
0	1	1	1
1	0	1	1
1	1	0	0

Exemplu de funcție cu valori indiferente:

x_2	x_1	x_0	f
0	0	0	0
0	0	1	0
0	1	0	X
0	1	1	1
1	0	0	X
1	0	1	1
1	1	0	0
1	1	1	X

Exprimări simbolice ale funcției:

$$f = R_0(0,1,6) + R_x(2,4,7) \quad \text{sau} \quad f = R_1(3,5) + R_x(2,4,7)$$

ESSC curs 3

Implementarea operatorilor logici

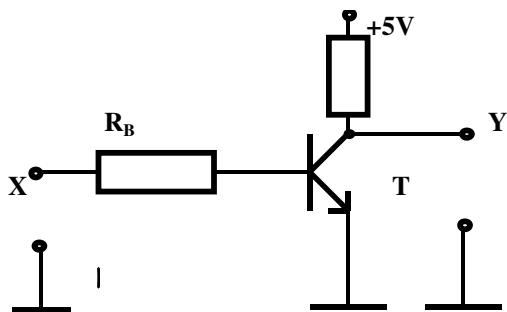
Calculatoarele electronice digitale lucrează cu impulsuri electrice standardizate în amplitudine și durată. Este deci posibil ca șirurile de impulsuri – care se succed la intervale egale de timp – să fie reprezentate prin șiruri de numere, notându-se prin *1* prezența impulsurilor și prin *0* absența lor.

Pe de altă parte, se poate considera că întregul sistem de calcul, ca și fiecare dintre subsistemele sale în parte, este reprezentat de către o funcție booleană pentru care se cunosc atât variabilele de intrare cât și valorile corespunzătoare pe care aceasta le iau.

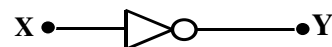
Concluzia este că orice subansamblu al calculatorului poate fi obținut prin configurarea convenabilă a unor circuite a căror funcționare poate fi descrisă prin intermediul operatorilor logici fundamentali: *AND*, *OR* și *NOT*. Rezultă de aici necesitatea studierii realizării unor circuite logice elementare care să implementeze operatorii logici fundamentali.

1 Operatorul de negare (complementare)

Acest circuit trebuie să funcționeze astfel încât atunci când la intrare se aplică un semnal *1* logic - spre exemplu un nivel de tensiune de *+5V* - în ieșire să apară un semnal *0* logic - spre exemplu *0V* - și invers. Circuitul trebuie deci să prezinte o intrare și o ieșire. Un astfel de circuit poate fi realizat după schema următoare:

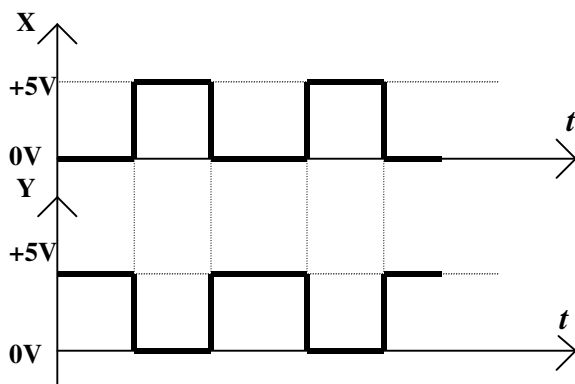


Simbolul:



Denumirea: NOT

Funcția: $Y = \bar{X}$

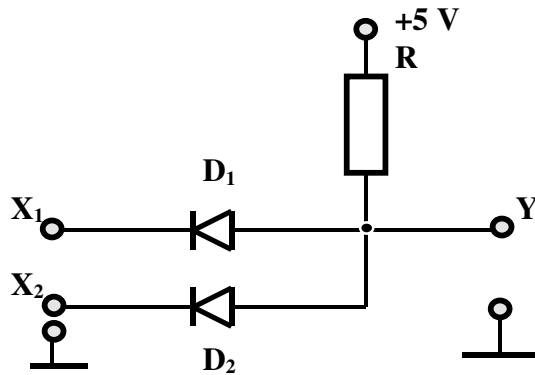


NOT	\bar{X}	$Y=X$
0	0	1
1	1	0

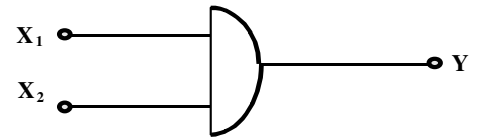
2 Operatorul de conjuncție (ȘI logic)

Circuitul se caracterizează prin două borne de intrare și o singură ieșire.

Tabela de adevăr afirmă că în ieșire trebuie să se găsească un semnal 1 logic (+5V) numai în situația când ȘI pe prima intrare ȘI pe cea de-a doua se aplică semnale 1 logic.

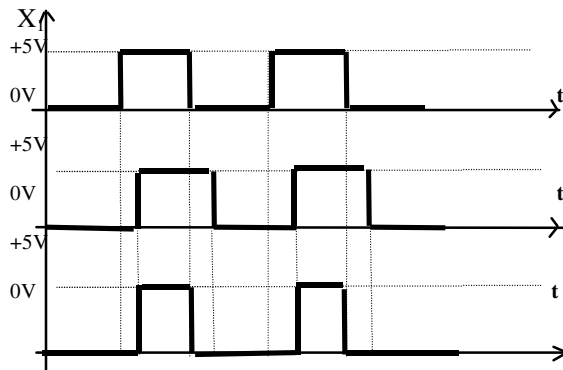


Simbolul:



Denumirea: AND

Funcția: $Y = X_1 \cdot X_2$



NOT	\bar{X}	Y=X
0	0	1
1	1	0