

Anatol GREMALSCHI
Iurie MOCANU
Ludmila GREMALSCHI

INFORMATICĂ

Manual pentru clasa a **10**-a

Elaborat conform curriculumului disciplinar în vigoare și aprobat prin Ordinul ministrului educației (nr. 211 din 11 aprilie 2012). Editat din sursele financiare ale *Fondului Special pentru Manuale*.

Comisia de experți: *Teodora Gherman*, dr. în pedagogie, conferențiar, șef Catedră Tehnologii Informaționale Aplicate, Academia de Administrare Publică de pe lângă Președintele Republicii Moldova; *Gheorghe Chistruga*, prof. șc., grad did. superior, Liceul Teoretic „Mihai Eminescu”, Drochia; *Mihai Chigai*, prof. șc., grad did. I, Liceul Teoretic Caplani, rn. Ștefan-Vodă

Recenzenți: *Gheorghe Ciocanu*, dr. habilitat în informatică, profesor universitar, Universitatea de Stat din Moldova; *Valeriu Cabac*, dr. în fizică și matematică, conferențiar universitar, Universitatea de Stat „Alecu Russo”, Bălți; *Mihai Șleahțișchi*, dr. în psihologie și pedagogie, conferențiar universitar, Universitatea Liberă Internațională din Moldova; *Tatiana Cartaleanu*, dr. în filologie, conferențiar universitar, Universitatea Pedagogică de Stat „Ion Creangă”, Chișinău; *Alexei Colîbneac*, Maestru în Arte, profesor universitar, Academia de Muzică, Teatru și Arte Plastice, Chișinău

Redactor: *Vasile Bahnaru*

Corectori: *Mariana Belenciuc, Oana Stoian*

Redactor tehnic: *Nina Duduciuc*

Machetare computerizată: *Anatol Andrișchi*

Copertă: *Vitaliu Pogolșa*

Întreprinderea Editorial-Poligrafică Știința,

str. Academiei, nr. 3; MD-2028, Chișinău, Republica Moldova;
tel.: (+373 22) 73-96-16; fax: (+373 22) 73-96-27;
e-mail: prini@stiinta.asm.md

DIFUZARE:

ÎM Societatea de Distribuție a Cărții *PRO-NOI*

str. Alba-Iulia, nr. 23/1A; MD-2051, Chișinău;
tel.: (+373 22) 51-68-17, 51-57-49; fax: (+373 22) 50-15-81;
e-mail: info@pronoi.md; www.pronoi.md

Toate drepturile asupra acestei ediții aparțin Întreprinderii Editorial-Poligrafice *Știința*.

Descrierea CIP a Camerei Naționale a Cărții

Gremalschi, Anatol

Informatică: Man. pentru clasa a 10-a / Anatol Gremalschi, Iurie Mocanu, Ludmila Gremalschi; Min. Educației al Rep. Moldova. – Ch.: Î.E.P. *Știința*, 2012 (Tipogr. „SEREBIA” SRL). – 188 p.

ISBN 978-9975-67-818-6
004(075.3)

© *Anatol Gremalschi, Iurie Mocanu, Ludmila Gremalschi*. 2007, 2012

© Întreprinderea Editorial-Poligrafică *Știința*. 2007, 2012

CUPRINS

Conținuturi	Umanist	Real	Pagina
Introducere			5
1. TIPURI DE DATE STRUCTURATE			
1.1. Tipuri de date <i>tablou</i> (array)	•	•	7
1.2. Tipuri de date <i>șir de caractere</i>	•	•	14
1.3. Tipuri de date <i>articol</i> (record)	•	•	18
1.4. Instrucțiunea <i>with</i>	•	•	23
1.5. Tipuri de date <i>mulțime</i> (set)	•	•	26
1.6. Generalități despre fișiere	•	•	31
1.7. Fișiere secvențiale	•	•	34
1.8. Fișiere <i>text</i>	•	•	38
<i>Test de autoevaluare nr. 1</i>	•	•	44
2. INFORMAȚIA			
2.1. Cantitatea de informație	•	•	46
2.2. Codificarea și decodificarea informației	•	•	49
2.3. Coduri frecvent utilizate	•	•	51
2.4. Informația mesajelor continue	•	•	56
2.5. Cuantizarea imaginilor	•	•	60
2.6. Reprezentarea și transmiterea informației	•	•	62
<i>Test de autoevaluare nr. 2</i>	•	•	66
3. BAZELE ARITMETICE ALE TEHNICII DE CALCUL			
3.1. Sisteme de numerație	•	•	68
3.2. Conversiunea numerelor dintr-un sistem în altul	•	•	71
3.3. Conversiunea din binar în octal, hexazecimal și invers		•	73
3.4. Operații aritmetice în binar		•	76
3.5. Reprezentarea numerelor naturale în calculator		•	78
3.6. Reprezentarea numerelor întregi		•	79
3.7. Reprezentarea numerelor reale		•	82
<i>Test de autoevaluare nr. 3</i>		•	86
4. ALGEBRA BOOLEANĂ			
4.1. Variabile și expresii logice		•	88
4.2. Funcții logice		•	91
4.3. Funcții logice frecvent utilizate		•	94
<i>Test de autoevaluare nr. 4</i>		•	96

Conținuturi	Umanist	Real	Pagina
5. CIRCUITE LOGICE			
5.1. Circuite logice elementare		•	98
5.2. Clasificarea circuitelor logice		•	103
5.3. Sumatorul		•	103
5.4. Circuite combinaționale frecvent utilizate		•	107
5.5. Bistabilul RS		•	110
5.6. Circuite secvențiale frecvent utilizate		•	113
5.7. Generatoare de impulsuri		•	116
<i>Test de autoevaluare nr. 5</i>		•	118
6. STRUCTURA ȘI FUNCȚIONAREA CALCULATORULUI			
6.1. Schema funcțională a calculatorului	•	•	121
6.2. Formatul instrucțiunilor		•	123
6.3. Tipuri de instrucțiuni		•	126
6.4. Limbajul cod calculator și limbajul de asamblare		•	127
6.5. Resursele tehnice și resursele programate ale calculatorului	•	•	130
6.6. Memorii externe pe benzi și discuri magnetice	•	•	131
6.7. Memorii externe pe discuri optice	•	•	135
6.8. Vizualizatorul și tastatura	•	•	139
6.9. Imprimantele	•	•	141
6.10. Clasificarea calculatoarelor	•	•	144
6.11. Microprocesorul		•	145
<i>Test de autoevaluare nr. 6</i>	•	•	147
7. REȚELE DE CALCULATOARE			
7.1. Introducere în rețele	•	•	150
7.2. Tehnologii de cooperare în rețea	•	•	153
7.3. Topologia și arhitectura rețelelor		•	155
7.4. Rețeaua <i>Internet</i>	•	•	158
7.5. Servicii <i>Internet</i>	•	•	163
<i>Test de autoevaluare nr. 7</i>	•	•	168
Răspunsuri la testele de autoevaluare			170
Bibliografie			187

INTRODUCERE

Impresionantele realizări în domeniul informaticii, crearea supercalculatoarelor și a calculatoarelor personale, apariția ciber spațiului, a realității virtuale și a Internetului presupun o cunoaștere profundă a principiilor de funcționare și a structurii calculatoarelor moderne. Din motive justificate, aceste principii au devenit o călăuză demnă de toată încrederea într-o lume aflată mereu în schimbare.

Manualul de față are drept scop însușirea de către elevi a cunoștințelor necesare pentru prelucrarea automată a informației cu ajutorul calculatoarelor numerice.

Capitolul 1 include un amplu material teoretic și practic referitor la definirea și prelucrarea datelor structurate: a tablourilor, a șirurilor de caractere, a articolelor, a mulțimilor și fișierelor. Sînt prezentate metodele de creare și de prelucrare a fișierelor: asocierile fișierelor PASCAL cu fișiere externe, scrierea și citirea componentelor unui fișier.

Capitolul 2 include expunerea unor cunoștințe fundamentale din teoria informației, și anume: cantitatea de informație din mesajele continue și discrete, codificarea și decodificarea informației, prezentarea informației în calculator.

În **Capitolul 3** sînt expuse cunoștințe fundamentale din domeniul aritmeticii sistemelor de calcul: sisteme de numerație și operații aritmetice în sistemul binar, reprezentarea numerelor naturale, numerelor întregi și a numerelor reale în calculator.

Capitolul 4 include unele cunoștințe fundamentale din domeniul algebrei booleene. Sînt expuse noțiunile de variabilă, constantă și funcții booleene, se examinează funcțiile logice frecvent utilizate.

În **Capitolul 5** se studiază circuitele logice ale unui calculator numeric: sumatorul, comparatorul, codificatorul și decodificatorul, bistabilul, registrul, numărătorul, generatorul de impulsuri.

Structura și funcționarea calculatorului sînt descrise în **Capitolul 6**. Materialul este expus în așa mod încît structura unui calculator numeric poate fi înțeleasă și însușită metodic, de la porți logice, dispozitive și unități la sisteme de calcul. O atenție deosebită se acordă interdependenței dintre conceptele matematice și realizarea fizică a echipamentelor unui sistem de calcul, interconexiunilor dintre resursele tehnice și cele programate ale calculatorului.

În **Capitolul 7** se studiază rețelele de calculatoare. Sînt expuse tehnologiile de cooperare în rețea, topologia și arhitectura rețelelor locale, regionale și globale. Pentru a facilita explorarea ciber spațiului, acest capitol include cunoștințe fundamentale despre Internet și serviciile din rețea: transferul fișierelor, poșta electronică, paginile Web.

Manualul este realizat în conformitate cu *Curriculumul disciplinar de informatică pentru învățămîntul liceal*, aprobat prin Ordinul Ministerului Educației al Republicii Moldova nr. 244 din 27 aprilie 2010. Repartizarea materialului pe profiluri – umanist și real – este reflectată în cuprinsul manualului.

Capitolul 1

TIPURI DE DATE STRUCTURATE

1.1. Tipuri de date *tablou* (array)

Mulțimea de valori ale unui tip de date **array** este constituită din tablouri (tabele). Tablourile sînt formate dintr-un număr fixat de componente de același tip, denumit **tip de bază**. Referirea componentelor se face cu ajutorul unui **indice**.

Un tip de date *tablou* se definește printr-o construcție de forma

```
type <Nume tip> = array [ $T_1$ ] of  $T_2$ ;
```

unde T_1 este tipul indicelui care trebuie să fie ordinal, iar T_2 este tipul componentelor (tipul de bază) care poate fi un tip oarecare.

Exemple:

- 1)

```
type Vector = array [1..5] of real;  
var x : Vector;
```
- 2)

```
type Zi = (L, Ma, Mi, J, V, S, D);  
Venit = array [Zi] of real;  
var v : Venit;  
z : Zi;
```
- 3)

```
type Ora = 0..23;  
Grade = -40..40;  
Temperatura = array [Ora] of Grade;  
var t : Temperatura;  
h : Ora;
```

Structura datelor din exemplele în studiu este prezentată în *figura 1.1*.

Fiecare componentă a unei variabile de tip *tablou* poate fi specificată explicit, prin numele variabilei urmat de indicele respectiv încadrat de paranteze pătrate.

Exemple:

- 1)

```
x[1], x[4];
```
- 2)

```
v[L], v[Ma], v[J];
```
- 3)

```
t[0], t[15], t[23];
```
- 4)

```
v[z], t[h].
```

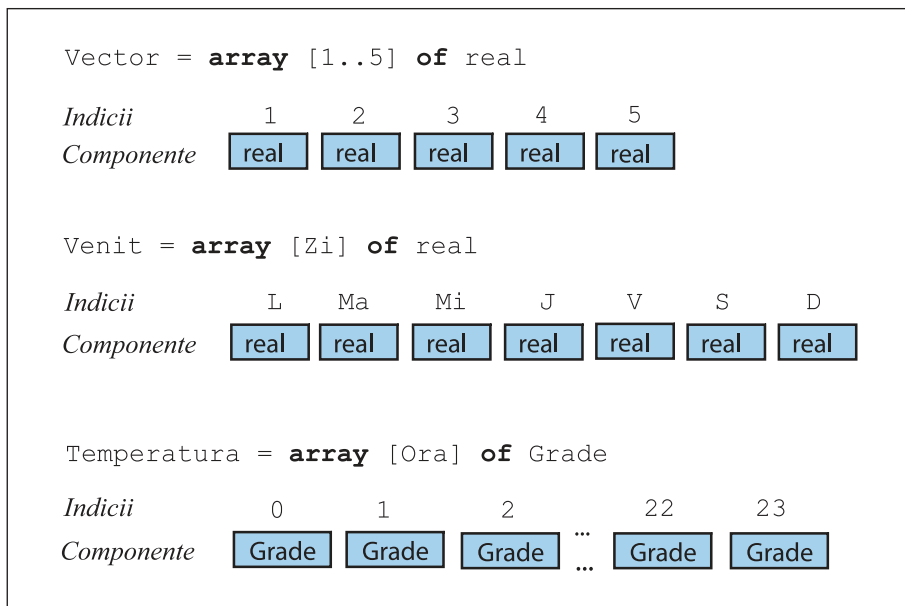


Fig. 1.1. Structura datelor de tip Vector, Venit și Temperatura

Asupra componentelor datelor de tip *tablou* se pot efectua toate operațiile admise de tipul de bază respectiv. Programul ce urmează afișează pe ecran suma componentelor variabilei `x` de tip `Vector`. Valorile componentelor `x[1]`, `x[2]`, ..., `x[5]` se citesc de la tastatură.

```
Program P77;
{ Suma componentelor variabilei x de tip Vector }
type Vector = array [1..5] of real;
var x : Vector;
    i : integer;
    s : real;
begin
  writeln('Dati 5 numere:');
  for i:=1 to 5 do readln(x[i]);
  writeln('Ati introdus:');
  for i:=1 to 5 do writeln(x[i]);
  s:=0;
  for i:=1 to 5 do s:=s+x[i];
  writeln('Suma=', s);
  readln;
end.
```

Pentru a extinde aria de aplicare a unui program, se recomandă ca numărul de componente ale datelor de tip **array** să fie specificate prin constante.

De exemplu, programul P77 poate fi modificat pentru a însuma n numere reale, $n \leq 100$:


```

Program P78;
{ Extinderea domeniului de aplicare a programului P77 }
const nmax = 100;
type Vector = array [1..nmax] of real;
var x : Vector;
    n : 1..nmax;
    i : integer;
    s : real;
begin
  write('n='); readln(n);
  writeln('Dati ', n, ' numere:');
  for i:=1 to n do readln(x[i]);
  writeln('Ati introdus:');
  for i:=1 to n do writeln(x[i]);
  s:=0;
  for i:=1 to n do
    s:=s+x[i];
  writeln('Suma=', s);
  readln;
end.

```

Tablourile bidimensionale se definesc cu ajutorul construcției

```

type <Nume tip> = array [ $T_1, T_2$ ] of  $T_3$ ;

```

unde T_1 și T_2 specifică tipul indicilor, iar T_3 – tipul componentelor.

Pentru exemplificare, în *figura 1.2* este prezentată structura datelor tipului:

```

Matrice = array [1..3, 1..4] of real

```

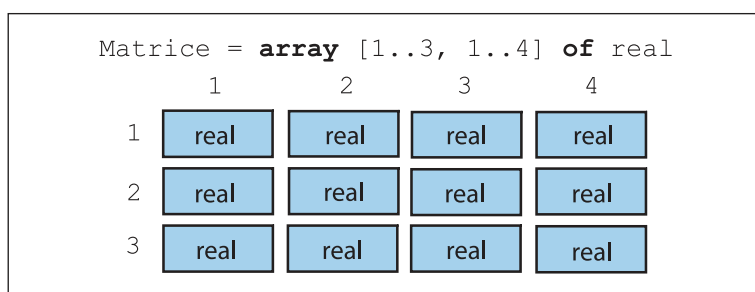


Fig. 1.2. Structura datelor de tip Matrice

Componentele unei variabile de tip *tablou bidimensional* se specifică explicit prin numele variabilei urmat de indicii respectivi separați prin virgulă și încadrați de paranteze pătrate.

De exemplu, în prezența declarației

```

var m : Matrice;

```

notația $m[1, 1]$ specifică componenta din linia 1, coloana 1 (vezi *fig. 1.2*); notația $m[1, 2]$ specifică componenta din linia 1, coloana 2; notația $m[i, j]$ specifică componenta din linia i , coloana j .

Programul ce urmează afișează pe ecran suma componentelor variabilei m de tip *Matrice*. Valorile componentelor $m[1, 1]$, $m[1, 2]$, ..., $m[3, 4]$ se citesc de la tastatură.

```
Program P79;
{ Suma componentelor variabilei m de tip Matrice }
type Matrice = array [1..3, 1..4] of real;
var m : Matrice;
    i, j : integer;
    s : real;
begin
  writeln('Dati componentele m[i,j]:');
  for i:=1 to 3 do
    for j:=1 to 4 do
      begin
        write('m[', i, ', ', j, ']=');
        readln(m[i,j]);
      end;
  writeln('Ati introdus:');
  for i:=1 to 3 do
    begin
      for j:=1 to 4 do write(m[i,j]);
      writeln;
    end;
    s:=0;
    for i:=1 to 3 do
      for j:=1 to 4 do
        s:=s+m[i,j];
      writeln('Suma=', s);
      readln;
    end.
end.
```

În general, un tip **tablou n -dimensional** ($n = 1, 2, 3$ etc.) se definește cu ajutorul diagramelor sintactice din *figura 1.3*. Atributul **packed** (împachetat) indică cerința de optimizare a spațiului de memorie pentru elementele tipului **array**. Menționăm că în majoritatea compilatoarelor actuale utilizarea acestui atribut nu are niciun efect, întrucât optimizarea se efectuează în mod automat.

Fiind date două variabile de tip *tablou* de același tip, numele variabilelor pot apărea într-o instrucțiune de atribuire. Această atribuire înseamnă copierea tuturor componentelor din membrul drept în cel stâng.

De exemplu, în prezența declarațiilor

```
var a, b : Matrice;
```

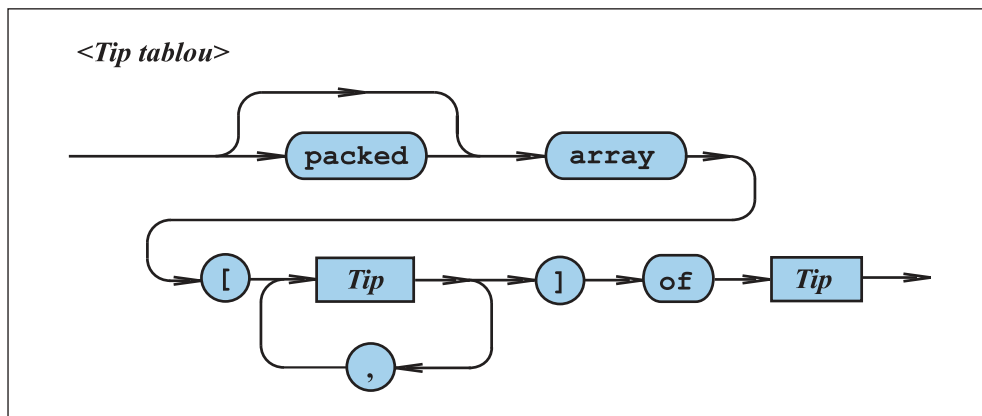


Fig. 1.3. Diagrama sintactică <Tip tablou>

instrucțiunea

```
a:=b
```

este corectă.

În exemplele de mai sus tipul de bază (tipul componentelor) a fost de fiecare dată un tip simplu. Deoarece tipul de bază poate fi, în general, un tip arbitrar, devine posibilă definirea tablourilor cu componente de tip structurat. Considerăm acum un exemplu în care tipul de bază este el însuși un tip **array**:

```
Type Linie = array [1..4] of real;
      Tabel = array [1..3] of Linie;
var L : Linie;
    T : Tabel;
    x : real;
```

Variabila T este formată din 3 componente: T[1], T[2] și T[3] de tipul Linie. Variabila L este formată din 4 componente: L[1], L[2], L[3] și L[4] de tipul real.

Prin urmare, atribuirile

```
L[1]:=x; x:=L[3]; T[2]:=L; L:=T[1]
```

sînt corecte.

Elementele variabilei T pot fi specificate prin T[i][j] sau prescurtat T[i, j]. Aici i indică numărul componentei de tip Linie în cadrul variabilei T, iar j – numărul componentei de tip real în cadrul componentei T[i] de tip Linie.

Subliniem faptul că declarațiile de forma

```
array [T1, T2] of T3
```

și

```
array [T1] of array [T2] of T3
```

definesc tipuri distincte de date. Prima declarație definește tablouri bidimensionale cu componente de tipul T_3 . A doua declarație definește tablouri unidimensionale cu componente de tipul **array** [T_2] **of** T_3 .

În programele PASCAL tablourile se utilizează pentru a grupa sub un singur nume mai multe variabile cu caracteristici identice.

Întrebări și exerciții

- ❶ Precizați tipul indicilor și tipul componentelor din următoarele declarații:

```
type P = array [1..5] of integer;
      Culoare = (Galben, Verde, Albastru, Violet);
      R = array [Culoare] of real;
      S = array [Culoare, 1..3] of boolean;
      T = array [boolean] of Culoare;
```

Reprezentați structura datelor de tipul P, R, S și T pe un desen (*fig. 1.1 și 1.2*).

- ❷ Indicați pe diagrama sintactică din *figura 1.3* drumurile care corespund declarațiilor din exercițiul 1.
- ❸ Scrieți formulele metalingvistice care corespund diagramei sintactice <Tip tablou> din *figura 1.3*.
- ❹ Se consideră declarațiile:

```
type Vector = array [1..5] of real;
var x, y : Vector;
```

Scrieți expresia aritmetică a cărei valoare este:

- a) suma primelor trei componente ale variabilei x;
 b) suma tuturor componentelor variabilei y;
 c) produsul tuturor componentelor variabilei x;
 d) valoarea absolută a componentei a treia a variabilei y;
 e) suma primelor componente ale variabilelor x și y.
- ❺ Se consideră declarațiile

```
type Zi = (L, Ma, Mi, J, V, S, D);
      Venit = array [Zi] of real;
var v : Venit;
```

Componentele variabilei v reprezintă venitul zilnic al unei întreprinderi. Elaborați un program care:

- a) calculează venitul săptămînal al întreprinderii;
 b) calculează media veniturii zilnice;
 c) indică ziua în care s-a obținut cel mai mare venit;
 d) indică ziua cu venitul cel mai mic.
- ❻ Se consideră declarațiile

```
type Ora = 0..23;
      Grade = -40..40;
      Temperatura = array [Ora] of Grade;
var t : Temperatura;
```

Componentele variabilei *t* reprezintă temperaturile măsurate din oră în oră pe parcursul a 24 de ore. Elaborați un program care:

- a) calculează temperatura medie;
- b) indică maximum și minimum temperaturii;
- c) indică ora (orele) la care s-a înregistrat temperatura maximă;
- d) indică ora (orele) la care s-a înregistrat temperatura minimă.

7 Se consideră declarațiile

```
type Oras = (Chisinau, Orhei, Balti, Tighina, Tiraspol);
      Zi = (L, Ma, Mi, J, V, S, D);
      Consum = array [Oras, Zi] of real;
var C : Consum;
      r : Oras;
      z : Zi;
```

Componenta *C[r, z]* a variabilei *C* reprezintă consumul de energie electrică a orașului *r* în ziua *z*. Elaborați un program care:

- a) calculează energia electrică consumată de fiecare oraș pe parcursul unei săptămîni;
- b) calculează energia electrică consumată zilnic de orașele în studiu;
- c) indică orașul cu un consum săptămînal maxim;
- d) indică orașul cu un consum săptămînal minim;
- e) indică ziua în care orașele consumă cea mai multă energie electrică;
- f) indică ziua în care orașul consumă cea mai puțină energie electrică.

8 Se consideră declarațiile

```
type Vector = array [1..5] of real;
      Matrice = array [1..3, 1..4] of real;
      Linie = array [1..4] of real;
      Tabel = array [1..3] of Linie;
var V : Vector;
      M : Matrice;
      L : Linie;
      T : Tabel;
      x : real;
      i : integer;
```

Care dintre atribuirile ce urmează sînt corecte?

a) *T[3] := T[1];*

h) *i := M[1, 2];*

b) *M := T;*

i) *x := V[4];*

c) *L := V;*

j) *L[3] := V[4];*

d) *L[3] := x;*

k) *T[1] := 4;*

e) *x := i;*

l) *T[2] := V;*

f) *i := x;*

m) *L := T[3];*

g) *L[3] := i;*

n) *T[1, 2] := M[1, 2];*

o) `T[1, 2] := M[1, 2];`

s) `x := M[1];`

p) `M[1] := 4;`

t) `L := M[1];`

q) `M[1, 3] := L[2];`

u) `V[5] := M[3, 4];`

r) `x := T[1][2];`

v) `L := M[3, 4].`

- 9 Utilizînd un tip de date *tablou*, elaborați un program care realizează algoritmul lui Eratostene pentru calcularea numerelor prime mai mici decît un număr natural dat n ($n \leq 200$).

1.2. Tipuri de date *șir de caractere*

În limbajul-standard tipul de date *șir de caractere* reprezintă un caz special al tipului **array** și se definește printr-o construcție de forma

`<Nume tip> ::= packed array [1..n] of char;`

Mulțimea de valori ale tipului de date în studiu este formată din toate șirurile ce conțin exact n caractere.

Exemplu:

```
Program P80;  
{ Siruri de caractere de lungime constanta }  
type Nume = packed array [1..8] of char;  
      Prenume = packed array [1..5] of char;  
var N : Nume;  
     P : Prenume;  
begin  
  N := 'Munteanu';  
  P := 'Mihai';  
  writeln(N);  
  writeln(P);  
  readln;  
end.
```

Rezultatul afișat pe ecran:

```
Munteanu  
Mihai
```

Întrucît șirurile de lungime diferită aparțin unor tipuri distincte de date, în cadrul programului P80 nu sînt admise atribuiri de genul:

```
N := 'Olaru';  
P := 'Ion'.
```

În astfel de cazuri, programatorul va completa datele respective cu spațiul pînă la numărul stabilit de caractere n , de exemplu:

```
N:= 'Olaru ';  
P:= 'Ion '.
```

Valorile unei variabile v de tip **packed array** $[1..n]$ of char pot fi introduse de la tastatură numai prin citirea separată a componentelor respective:

```
read(v[1]); read(v[2]); ...; read(v[n]).
```

În schimb, o astfel de valoare poate fi afișată în totalitatea ei printr-un singur apel `write(v)` sau `writeln(v)`.

Subliniem faptul că șirurile de caractere de tip **packed array** $[1..n]$ of char conțin exact n caractere, adică sînt **șiruri de lungime constantă**. Evident, lungimea lor nu poate fi modificată pe parcursul derulării programului respectiv. Acest fapt complică elaborarea programelor destinate prelucrării unor șiruri arbitrare de caractere.

Pentru a elimina acest neajuns, versiunile actuale ale limbajului permit utilizarea șirurilor de caractere de lungime variabilă. În Turbo PASCAL un tip de date **șir de caractere de lungime variabilă** se declară printr-o construcție de forma:

```
type <Nume tip> = string;
```

sau

```
type <Nume tip> = string [ $nmax$ ];
```

unde $nmax$ este lungimea maximă pe care o pot avea șirurile respective. În lipsa parametrului $nmax$ lungimea maximă se stabilește implicit, în mod obișnuit – 255 de caractere.

Asupra șirurilor de tip **string** se poate efectua operația de concatenare (juxtapunere), notată prin semnul „+”. Lungimea curentă a unei valori v de tip **string** poate fi aflată cu ajutorul funcției predefinite `length(v)` care returnează o valoare de tip **integer**. Indiferent de lungime, toate șirurile de caractere de tip **string** sînt compatibile.

Exemplu:

```
Program P81;  
{ Siruri de caractere de lungime variabila }  
type Nume = string [8];  
    Prenume = string [5];  
    NumePrenume = string;  
var N : Nume;  
    P : Prenume;  
    NP : NumePrenume;  
    L : integer;
```

begin

```
N:='Munteanu';   L:=length(N);   writeln(N, L:4);
P:='Mihai';      L:=length(P);   writeln(P, L:4);
NP:=N+' '+P;     L:=length(NP);  writeln(NP, L:4);
N:='Olaru';       L:=length(N);   writeln(N, L:4);
P:='Ion';         L:=length(P);   writeln(P, L:4);
NP:=N+' '+P;     L:=length(NP);  writeln(NP, L:4);
readln;
```

end.

Rezultatele afișate pe ecran:

```
Munteanu   8
Mihai      5
Munteanu Mihai  14
Olaru      5
Ion        3
Olaru Ion   9
```

Se observă că pe parcursul derulării programului în studiu lungimea șirurilor de caractere N, P și NP se schimbă.

Asupra șirurilor de caractere sînt admise operațiile relaționale <, <=, =, >=, >, <>. Șirurile se compară componentă cu componentă de la stînga la dreapta în conformitate cu ordonarea caracterelor în tipul de date **char**. Ambii operanzi trebuie să fie de tip **packed array [1..n] of char** cu același număr de componente sau de tip **string**. Evident, operanzii de tip **string** pot avea lungimi arbitrare.

De exemplu, rezultatul operației

```
'AC' < 'BA'
```

este **true**, iar rezultatul operației

```
'AAAAC' < 'AAAAB'
```

este **false**.

O variabilă de tip *șir de caractere* poate fi folosită fie în totalitatea ei, fie parțial, prin referirea unui caracter din șir.

De exemplu, pentru P='Mihai' avem P[1]='M', P[2]='i', P[3]='h' ș.a.m.d. După executarea secvenței de instrucțiuni

```
P[1]:='P';
P[2]:='e';
P[3]:='t';
P[4]:='r';
P[5]:='u'
```

variabila P va avea valoarea 'Petru'.

Programul ce urmează citește de la tastatură șiruri arbitrare de caractere și afișează pe ecran numărul de spații în șirul respectiv. Derularea programului se termină după introducerea șirului 'Sfirsit'.

Program P82;

```
{ Numarul de spatii intr-un sir de caractere }  
var S : string;  
    i, j : integer;  
begin  
    writeln('Dati siruri de caractere:');  
    repeat  
        readln(S);  
        i:=0;  
        for j:=1 to length(S) do  
            if S[j]=' ' then i:=i+1;  
        writeln('Numarul de spatii=', i);  
    until S='Sfirsit';  
end.
```

Întrebări și exerciții

- ❶ Cum se definește un tip de date *șir de caractere*?
- ❷ Ce operații pot fi efectuate asupra șirurilor de caractere?
- ❸ Comentați următorul program:

Program P83;

```
{ Eroare }  
var S : packed array [1..5] of char;  
begin  
    S:='12345';  
    writeln(S);  
    S:='Sfat';  
    writeln(S);  
end.
```

- ❹ Elaborați un program care:
 - a) determină numărul de apariții ale caracterului 'A' într-un șir;
 - b) substituie caracterul 'A' prin caracterul '*';
 - c) radiază din șir caracterul 'B';
 - d) determină numărul de apariții ale silabei 'MA' într-un șir;
 - e) substituie silabele 'MA' prin silaba 'TA';
 - f) radiază din șir silaba 'TO'.
- ❺ Precizați rezultatul operațiilor relaționale:
 - a) 'B' < 'A';
 - b) 'BB' > 'AA';
 - c) 'BAAAA' < 'AAAAA';
 - d) 'CCCCD' > 'CCCCA';

e) 'A A' = 'AA';

h) 'Aa' > 'aA';

f) 'BB ' < 'B B';

i) '123' = '321';

g) 'A' = 'a';

j) '12345' > '12345'.

- ⑥ Se consideră șiruri de caractere formate din literele mari ale alfabetului latin și spații. Elaborați un program care afișează șirurile în studiu după următoarele reguli:
- fiecare literă de la 'A' pînă la 'Y' se înlocuiește prin următoarea literă din alfabet;
 - fiecare literă 'Z' se înlocuiește prin litera 'A';
 - fiecare spațiu se înlocuiește prin ' - '.
- ⑦ Elaborați un program care descifrează șirurile cifrate conform regulilor din exercițiul 6.
- ⑧ Se consideră m , $m \leq 100$ șiruri de caractere formate din literele mici ale alfabetului latin. Elaborați un program care afișează pe ecran șirurile în studiu în ordine alfabetică.
- ⑨ Șirul S este compus din cîteva propoziții, fiecare terminîndu-se cu punct, semn de exclamare sau semnul întrebării. Elaborați un program care afișează pe ecran numărul de propoziții din șirul în studiu.

1.3. Tipuri de date *articol* (record)

Mulțimea de valori ale unui tip de date **record** este constituită din articole (înregistrări). Articolele sînt formate din componente, denumite **cîmpuri**. Spre deosebire de componentele unui tablou, cîmpurile pot fi de tipuri diferite. Fiecare cîmp are un nume (identificator de cîmp).

Un tip de date *articol* se definește printr-o structură de forma

```
type <Nume tip> = record
    <Nume cîmp 1> : T1;
    <Nume cîmp 2> : T2;
    ...
    <Nume cîmp n> : Tn;
end;
```

unde $T_1, T_2, \dots, T_n$ specifică tipul cîmpurilor respective. Tipul unui nume de cîmp este arbitrar, astfel un cîmp poate să fie la rîndul lui tot de tip *articol*. Prin urmare, se pot defini tipuri imbricate.

Example:

```
1) type Elev = record
    Nume : string;
    Prenume : string;
    NotaMedie : real;
end;
var E1, E2 : Elev;
```

```

2) type Punct = record
      x : real; { coordonata x }
      y : real; { coordonata y }
    end;
var P1, P2 : Punct;

3) type Triunghi = record
      A : Punct; { virful A }
      B : Punct; { virful B }
      C : Punct; { virful C }
    end;
var T1, T2, T3 : Triunghi;

```

Structura datelor din exemplele în studiu este prezentată în *figura 1.4*.

Fiind date două variabile de tip *articol* de același tip, numele variabilelor pot apărea într-o instrucțiune de atribuire. Această atribuire înseamnă copierea tuturor câmpurilor din membrul drept în membrul stâng. De exemplu, pentru tipurile de date și variabilele declarate mai sus instrucțiunile

```

E1:=E2;
T2:=T3;
P2:=P1

```

sînt corecte.

Fiecare componentă a unei variabile de tip **record** poate fi specificată explicit, prin numele variabilei și denumirile de câmpuri, separate prin puncte.

Exemple:

1) E1.Nume, E1.Prenume, E1.NotaMedie;

2) E2.Nume, E2.Prenume, E2.NotaMedie;

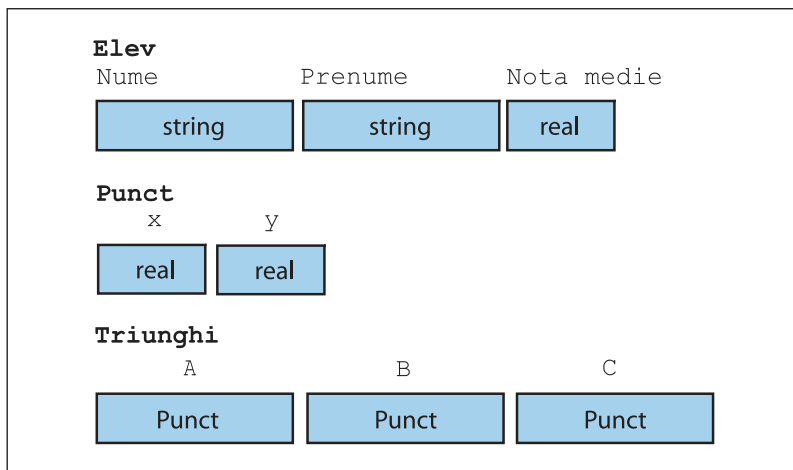


Fig. 1.4. Structura datelor de tip Elev, Punct și Triunghi

- 3) P1.x, P1.y, P2.x, P2.y;
- 4) T1.A, T1.B, T1.C, T2.A, T2.B, T2.C;
- 5) T1.A.x, T1.A.y, T2.B.x, T2.B.y.

Evident, componenta E1.Nume este de tip **string**; componenta P1.x este de tip **real**; componenta T1.A este de tip **Punct**; componenta T1.A.x este de tip **real** ș.a.m.d.

Asupra componentelor datelor de tip *articol* se pot efectua toate operațiile admise de tipul câmpului respectiv. Programul ce urmează compară notele medii a doi elevi și afișează pe ecran numele și prenumele elevului cu nota medie mai bună. Se consideră că elevii au note medii diferite.

```

Program P84;
{ Date de tipul Elev }
type Elev = record
    Nume : string;
    Prenume : string;
    NotaMedie : real;
end;
var E1, E2, E3 : Elev;
begin
    writeln('Dati datele primului elev:');
    write('Nume:');    readln(E1.Nume);
    write('Prenume:'); readln(E1.Prenume);
    write('Nota medie:'); readln(E1.NotaMedie);

    writeln('Dati datele elevului al doilea:');
    write('Nume:');    readln(E2.Nume);
    write('Prenume:'); readln(E2.Prenume);
    write('Nota medie:'); readln(E2.NotaMedie);

    if E1.NotaMedie > E2.NotaMedie then E3:=E1 else E3:=E2;

    writeln('Elevul cu media mai buna:');
    writeln(E3.Nume, ' ', E3.Prenume, ':', E3.NotaMedie : 5:2);
    readln;
end.

```

Orice tip de date **record** poate servi ca tip de bază pentru formarea altor tipuri structurate.

Exemplu:

```

type ListaElevilor = array [1..40] of Elev;
var LE : ListaElevilor;

```

Evident, notația $LE[i]$ specifică elevul i din listă; notația $LE[i].Nume$ specifică numele acestui elev ș.a.m.d. Programul ce urmează citește de la tastatură datele referitoare la n elevi și afișează pe ecran numele, prenumele și nota medie a celui mai bun elev. Se consideră că elevii au note medii diferite.

```
Program P85;
{ Tablou cu componente de tipul Elev }
type Elev = record
    Nume : string;
    Prenume : string;
    NotaMedie : real;
end;
    ListaElev = array [1..40] of Elev;
var E : Elev;
    LE : ListaElev;
    n : 1..40;
    i : integer;
begin
    write('n='); readln(n);
    for i:=1 to n do
        begin
            writeln('Dati datele elevului ', i);
            write('Numele: '); readln(LE[i].Nume);
            write('Prenumele: ');
            readln(LE[i].Prenume);
            write('Nota Medie: ');
            readln(LE[i].NotaMedie);
        end;
        E.NotaMedie:=0;
        for i:=1 to n do
            if LE[i].NotaMedie > E.NotaMedie then E:=LE[i];
        writeln('Cel mai bun elev:');
        writeln(E.Nume, ' ', E.Prenume, ':', E.NotaMedie : 5:2);
        readln;
    end.
```

În general, un tip de date *articol* se definește cu ajutorul diagramelor sintactice din figura 1.5. În completare la articolele cu un număr fix de câmpuri, limbajul PASCAL permite utilizarea articolelor cu variante. Aceste tipuri de date se studiază în cursurile avansate de informatică.

Întrebări și exerciții

- 1 Care este mulțimea de valori ale unui tip de date *articol*?
- 2 Indicați pe diagrama sintactică din figura 1.5 drumurile care corespund definițiilor tipurilor de date *articol* din programele P84 și P85.

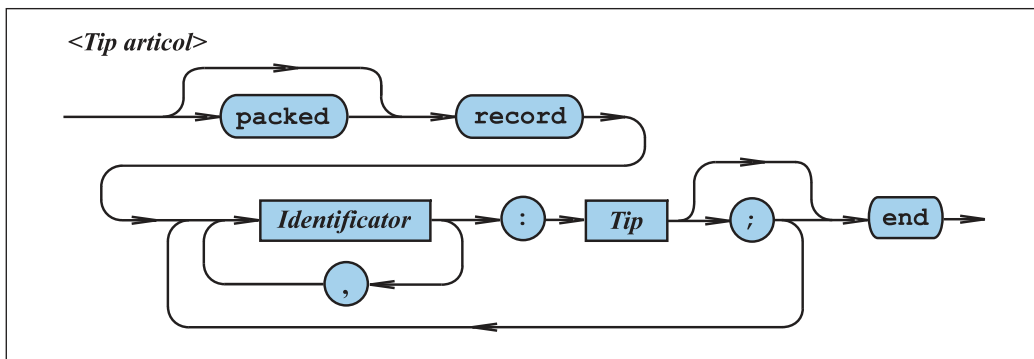


Fig. 1.5. Diagrama sintactică <Tip articol>

- ③ Scrieți formulele metalingvistice pentru diagrama sintactică din figura 1.5.
- ④ Se consideră următoarele tipuri de date

```

type Data = record
    Ziua : 1..31;
    Luna : 1..12;
    Anul : integer;
end;
Persoana = record
    NumePrenume : string;
    DataNasterii : Data;
end;
ListaPersoane = array [1..50] of Persoana;

```

Elaborați un program care citește de pe tastatură datele referitoare la n persoane ($n \leq 50$) și afișează pe ecran:

- a) persoanele născute în ziua z a lunii;
 - b) persoanele născute în luna l a anului;
 - c) persoanele născute în anul a ;
 - d) persoanele născute pe data $z.l.a$;
 - e) persoana cea mai în vârstă;
 - f) persoana cea mai tânără;
 - g) vârsta fiecărei persoane în ani, luni, zile;
 - h) lista persoanelor care au mai mult de v ani;
 - i) lista persoanelor în ordine alfabetică;
 - j) lista persoanelor ordonată conform datei nașterii;
 - k) lista persoanelor de aceeași vârstă (născuți în același an).
- ⑤ Se consideră n puncte ($n \leq 30$) pe un plan euclidian. Fiecare punct i este definit prin coordonatele sale. Distanța dintre punctele i și j se calculează după formula

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}.$$

Elaborați un program care afișează pe ecran punctele distanța dintre care este maximă.

- ⑥ Aria triunghiului este dată de formula lui Heron

$$S = \sqrt{p(p-a)(p-b)(p-c)},$$

unde p este semiperimetrul, iar a , b și c sînt lungimile laturilor respective. Utilizînd tipurile de date **Punct** și **Triunghi** din paragraful în studiu, elaborați un program care citește de la tastatură informațiile referitoare la n triunghiuri ($n \leq 10$) și afișează pe ecran:

- aria fiecărui triunghi;
- coordonatele vîrfurilor triunghiului cu aria maximă;
- coordonatele vîrfurilor triunghiului cu aria minimă;
- informațiile referitoare la fiecare triunghi în ordinea creșterii ariilor.

1.4. Instrucțiunea **with**

Componentele unei variabile de tip *articol* se specifică explicit prin numele variabilei și denumirile de cîmpuri, separate prin puncte.

De exemplu, în prezența declarațiilor

```
type Angajat = record
    NumePrenume : string;
    ZileLucrate : 1..31;
    PlataPeZi : real;
    PlataPeLuna : real;
end;
var A : Angajat;
```

componentele variabilei **A** se specifică prin **A.NumePrenume**, **A.ZileLucrate**, **A.PlataPeZi** și **A.PlataPeLuna**.

Întrucît numele **A** al variabilei de tip *articol* se repetă de mai multe ori, acest mod de referire a componentelor este, în anumite situații, incomod. Repetările obositoare pot fi evitate cu ajutorul instrucțiunii **with** (cu).

Sintaxa instrucțiunii în studiu este:

<Instrucțiune with> ::= with <Variabilă> {, <Variabilă>} do <Instrucțiune>

Diagrama sintactică este prezentată în figura 1.6.

În interiorul unei instrucțiuni **with** componentele uneia sau a mai multe variabile de tip *articol* pot fi referite folosind numai numele cîmpurilor respective.

Exemplu:

```
with A do PlataPeLuna:=PlataPeZi*ZileLucrate
```

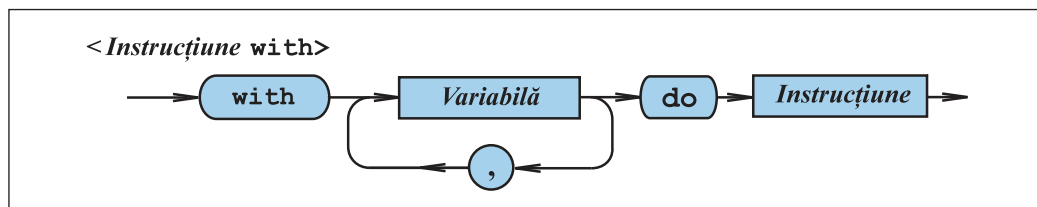


Fig. 1.6. Diagrama sintactică a instrucțiunii **with**

Această instrucțiune este echivalentă cu următoarea:

```
A.PlataPeLuna:=A.PlataPeZi*A.ZileLucrate
```

Utilizarea instrucțiunii **with** necesită o atenție sporită din partea programatorului, care este obligat să specifice univoc componentele variabilelor de tip *articol*. În interiorul unei instrucțiuni **with**, la întâlnirea unui identificator, prima dată se testează dacă el poate fi interpretat ca un nume de câmp al articolului respectiv. Dacă da, identificatorul va fi interpretat ca atare, chiar dacă în acel moment este accesibilă și o variabilă avînd același nume.

Exemplu:

```
type Punct = record
    x : real;
    y : real;
end;
Segment = record
    A : Punct;
    B : Punct;
end;
var P : Punct;
    S : Segment;
    x : integer;
```

În cazul nostru identificatorul *x* poate să reprezinte fie variabila *x* de tip *integer*, fie câmpul *P.x* al articolului *P*.

În instrucțiunea

```
x:=1
```

identificatorul *x* se referă la variabila *x* de tip *integer*.

În instrucțiunea

```
with P do x:=1
```

identificatorul *x* se referă la câmpul *P.x* al variabilei de tip articol *P*.

Întrucît variabila de tip articol *S* nu conține niciun câmp cu numele *S.x*, în instrucțiunea

```
with S do x:=1
```

identificatorul *x* va fi interpretat ca variabila *x* de tip *integer*.

O instrucțiune de forma

```
with  $v_1, v_2, \dots, v_n$  do <Instrucțiune>,
```

unde $v_1, v_2, \dots, v_n$ sînt variabile de tip *articol*, este echivalentă cu instrucțiunea

```
with  $v_1$  do
with  $v_2$  do
```



```
{...}
with  $v_n$  do <Instrucțiune>.
```

Evident, componentele variabilelor $v_1, v_2, \dots, v_n$ trebuie specificate univoc prin denumirile câmpurilor respective.

De exemplu, pentru variabilele P și S, declarate mai sus, putem scrie:

```
with P, S do
begin
  x:=1.0;    { referire la P.x }
  y:=1.0;
  A.x:=0;    { referire la S.A.x }
  A.y:=0;
  B.x:=2.0; { referire la S.B.x }
  B.y:=2.0;
end;
```

În mod obișnuit, instrucțiunea **with** se utilizează numai în cazurile în care se ajunge la o reducere semnificativă a textului unui program.

Întrebări și exerciții

- 1 Indicați pe diagrama sintactică din *figura 1.6* drumurile care corespund instrucțiunilor **with** din exemplele paragrafului în studiu.
- 2 Care este destinația instrucțiunii **with**?
- 3 Utilizînd instrucțiunea **with**, excludeți din programele P84 și P85 din paragraful precedent repetările de genul

```
E1.Nume, E1.Prenume, ...,
LE[i].Nume, LE[i].Prenume.
```

- 4 Se consideră următoarele tipuri de date:

```
type Angajat = record
    NumePrenume : string;
    ZileLucrate : 1..31;
    PlataPeZi : real;
    PlataPeLuna : real;
end;
ListaDePlata = array [1..50] of Angajat;
```

Plata pe lună a fiecărui angajat se calculează înmulțind plata pe zi cu numărul de zile lucrate. Elaborați un program care:

- a) calculează plata pe lună a fiecărui angajat;
- b) calculează salariul mediu al angajaților incluși în listă;
- c) afișează pe ecran datele despre angajații cu plata lunară maximă;
- d) afișează lista angajaților ordonată alfabetic;
- e) afișează lista angajaților în ordinea creșterii plăților pe zi;

- f) ordonează lista angajaților în ordinea creșterii plăților pe lună;
 - g) afișează lista angajaților în ordinea creșterii numărului de zile lucrate.
- 5 Un cerc poate fi definit prin coordonatele x, y și raza r . Elaborați un program care citește de la tastatură datele referitoare la n cercuri ($n \leq 50$) și afișează pe ecran:
- a) coordonatele centrului și raza cercului cu aria maximă;
 - b) numărul de cercuri incluse în cercul cu raza maximă și coordonatele centrelor respective;
 - c) coordonatele centrului și raza cercului cu aria minimă;
 - d) numărul de cercuri în care este inclus cercul cu raza minimă și coordonatele centrelor respective.

1.5. Tipuri de date *mulțime* (set)

Un tip de date *mulțime* (**set**) se definește în raport cu un tip de bază care trebuie să fie ordinal:

<Tip mulțime> ::= [packed] set of <Tip>

Valorile unui tip de date **set** sînt mulțimi formate din valorile tipului de bază. Dacă tipul de bază are n valori, tipul *mulțime* va avea 2^n valori. În implementările limbajului valoarea lui n este limitată, de regulă $n \leq 256$.

În PASCAL o mulțime poate fi specificată enumerîndu-i-se elementele între parantezele pătrate „[” și „]”, care țin locul acoladelor din matematică.

Notăția [] reprezintă mulțimea vidă.

Exemple:

```
type Indice = 1..10;
     Zi = (L, Ma, Mi, J, V, S, D);
     MultimeIndicii = set of Indice;
     ZileDePrezenta = set of Zi;
var MI : MultimeIndicii;
     ZP : ZileDePrezenta;
```

Tipul ordinal Indice are $n = 10$ valori: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10. Tipul MultimeIndicii are $2^{10} = 1\,024$ de valori, și anume:

```
[], [1], [2], ..., [1, 2], [1, 3], ...,
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10].
```

Prin urmare, variabila MI poate să aibă oricare din aceste valori, de exemplu:

```
MI:= [1, 3].
```

Tipul ordinal Zi are $n = 7$ valori: L, Ma, Mi, J, V, S, D. Tipul ZileDePrezenta are $2^7 = 128$ de valori, și anume:

```
[], [L], [Ma], [Mi], ..., [L, Ma], [L, Mi], ...,
[L, Ma, Mi, J, V, S, D].
```

Variabila ZP poate să aibă oricare dintre aceste valori, de exemplu,

ZP:= [L, Ma, Mi, V]

O valoare de tip *mulțime* poate fi specificată printr-un **constructor** (generator) de mulțime. Diagrama sintactică a unității gramaticale <Constructor mulțime> este prezentată în figura 1.7.

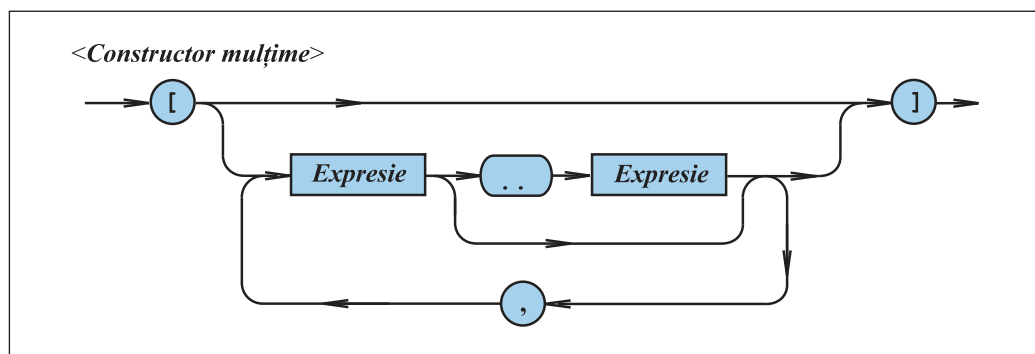


Fig. 1.7. Diagrama sintactică <Constructor mulțime>

Un constructor conține specificarea elementelor mulțimii, separate prin virgule și incluse între paranteze pătrate. Un element poate să fie o valoare concretă a tipului de bază sau un interval de forma:

<Expresie> .. <Expresie>.

Valorile expresiilor în studiu precizează limitele inferioare și superioare ale intervalului.

Exemple :

- 1) [];
- 2) [1, 2, 3, 8];
- 3) [1..4, 8..10];
- 4) [i-k..i+k];
- 5) [L, Ma, V..D].

Asupra valorilor unui tip de date *mulțime* se pot efectua operațiile uzuale:

- + reuniunea;
- * intersecția;
- diferența,

rezultatul fiind de tip *mulțime* și operațiile relaționale:

- = egalitatea;
- <> inegalitatea;

<=, >= incluziunea;
in apartenența,
rezultatul fiind de tip boolean.

Programul ce urmează afișează pe ecran rezultatele operațiilor +, * și -, efectuate asupra valorilor de tip MultimeIndicii.

```
Program P86;  
{ Date de tip MultimeIndicii }  
type Indice = 1..10;  
    MultimeIndicii = set of Indice;  
var A, B, C : MultimeIndicii;  
    i : integer;  
begin  
    A:= [1..5, 8];           { A contine 1, 2, 3, 4, 5, 8 }  
    B:= [1..3, 9, 10]; { B contine 1, 2, 3, 9, 10 }  
    C:= [];                  { C este o multime vida }  
  
    C:=A+B;                  { C contine 1, 2, 3, 4, 5, 8, 9, 10 }  
    writeln('Reuniune');  
    for i:=1 to 10 do  
        if i in C then write(i:3);  
    writeln;  
  
    C:=A*B;                  { C contine 1, 2, 3 }  
    writeln('Intersectie');  
    for i:=1 to 10 do  
        if i in C then write(i:3);  
    writeln;  
  
    C:=A-B;                  { C contine 4, 5, 8 }  
    writeln('Diferenta');  
    for i:=1 to 10 do  
        if i in C then write(i:3);  
    writeln;  
    readln;  
end.
```

Spre deosebire de tablouri și articole, componentele cărora pot fi referite direct, respectiv prin indicii și denumiri de cîmpuri, elementele unei mulțimi nu pot fi referite. Se admite numai verificarea apartenenței elementului la o mulțime (operația relațională **in**). În pofida acestui fapt, utilizarea tipurilor de date *mulțime* mărește viteza de execuție și îmbunătățește lizibilitatea programelor PASCAL. De exemplu, instrucțiunea:

```
if (c='A') or (c='E') or (c='I') or (c='O') or (c='U') then ...
```

poate fi înlocuită cu o instrucțiune mai simplă:

```
if c in ['A', 'E', 'I', 'O', 'U'] then ...
```

Un alt exemplu sugestiv este utilizarea tipurilor de date *mulțime* în calcularea numerelor prime mai mici decât un număr natural dat n . Pentru aceasta se folosește algoritmul *Ciurul (sita) lui Eratostene*:

- 1) în sită se depun numerele 2, 3, 4, ..., n ;
- 2) din sită se extrage cel mai mic număr i ;
- 3) numărul extras se include în mulțimea numerelor prime;
- 4) din sită se elimină toți multiplii m ai numărului i ;
- 5) procesul se încheie când sita s-a golit.

```
Program P87;  
{ Ciurul (sita) lui Eratostene }  
const n = 50;  
type MultimeDeNumere = set of 1..n;  
var Sita, NumerePrime : MultimeDeNumere;  
    i, m : integer;  
begin  
  {1} Sita:= [2..n];  
      NumerePrime:=[];  
      i:=2;  
      repeat  
  {2}   while not (i in Sita) do i:=succ(i);  
  {3}   NumerePrime:=NumerePrime+[i];  
        write(i:4);  
        m:=i;  
  {4}   while m<=n do  
        begin Sita:=Sita-[m]; m:=m+i; end;  
  {5} until Sita=[];  
        writeln;  
        readln;  
end.
```

Correspondența dintre punctele algoritmului și instrucțiunile care le exprimă este indicată în comentariile din partea stîngă a liniilor de program.

Întrebări și exerciții

- ❶ Enumerați valorile posibile ale variabilelor din declarațiile ce urmează:

```
var V : set of 'A'..'C';  
      S : set of (A, B, C);  
      I : set of '1'..'2';  
      J : set of 1..2;
```

- ② Comentați următorul program :

```
Program P88;  
{ Eroare }  
type Multime = set of integer;  
var M : Multime;  
    i : integer;  
begin  
    M:=[1, 8, 13];  
    for i:=1 to MaxInt do  
        if i in M then writeln(i);  
end.
```

- ③ Se consideră următoarele declarații:

```
type Culoare = (Galben, Verde, Albastru, Violet);  
    Nuanta = set of Culoare;  
var NT : Nuanta;
```

Care sînt valorile posibile ale variabilei NT?

- ④ Scrieți formula metalingvistică care corespunde diagramei sintactice <Construc-
tor mulțime> din figura 1.7.
- ⑤ Se consideră tipul de date MultimeIndicii din paragraful în studiu. Precizați
mulțimile specificate de constructorii ce urmează:
- | | |
|--------------------|---------------------|
| a) []; | f) [4..3]; |
| b) [1..10]; | g) [1..3, 7..6, 9]; |
| c) [1..3, 9..10]; | h) [4-2..7+1]; |
| d) [1+1, 4..7, 9]; | i) [7-5..4+4]; |
| e) [3, 7..9]; | j) [6, 9, 1..2]. |
- ⑥ Elaborați un program care afișează pe ecran toate submulțimile mulțimii {1, 2, 3, 4}.
- ⑦ Elaborați un program care afișează pe ecran toate submulțimile mulțimii
{‘A’, ‘B’, ‘C’, ‘D’}.
- ⑧ Se dă un șir de caractere în care cuvintele sînt separate fie prin spațiu, fie prin
caracterele *punct*, *virgulă*, *punct și virgulă*, *semnul exclamării* și *semnul între-*
bării. Elaborați un program care afișează pe ecran cuvintele șirului de caracte-
re citit de la tastatură.
- ⑨ Se dă un șir de caractere. Elaborați un program care afișează pe ecran numă-
rul de vocale din șir.
- ⑩ Elaborați un program care citește de la tastatură două șiruri de caractere și
afișează pe ecran:
- caracterele care se întîlnesc cel puțin în unul dintre șiruri;
 - caracterele care apar în ambele șiruri;
 - caracterele care apar în primul și nu apar în șirul al doilea.
- ⑪ Scrieți un program care verifică dacă numele unei persoane este introdus co-
rect (numele este un șir de caractere ce nu conține cifre).

- ❷ În implementările actuale ale limbajului numărul valorilor tipului de bază al unui tip *mulțime* este limitat, obișnuit $n \leq 256$. În consecință, programul P87 nu poate calcula numere prime mai mari decât n . Elaborați un program pentru calcularea numerelor prime din intervalul 8, ..., 10 000.

Indicație: Sita din algoritmul lui Eratostene poate fi reprezentată printr-un tablou, componentele căruia sînt mulțimi.

1.6. Generalități despre fișiere

Prin **fișier** se înțelege o structură de date care constă dintr-o secvență de componente. Fiecare componentă din secvență are același tip, denumit *tip de bază*. Numărul componentelor din secvență nu este fixat, însă sfîrșitul secvenței este indicat de un simbol special, notat *EOF* (*End of File* – sfîrșit de fișier). Fișierul care nu conține nicio componentă se numește *fișier vid*.

Un tip de date *fișier* se definește printr-o declarație de forma:

```
<Tip fișier> ::= [packed] file of <Tip>;
```

unde <Tip> este tipul de bază. Tipul de bază este un tip arbitrar, exceptînd tipul *fișier* (nu există „fișier de fișiere”).

Exemple:

```
1) type FisierNumere = file of integer;  
   var FN : FisierNumere;  
       n : integer;
```

```
2) type FisierCaractere = file of char;  
   var FC : FisierCaractere;  
       c : char;
```

```
3) type Elev = record  
        Nume : string;  
        Prenume : string;  
        NotaMedie : real;  
      end;  
   FisierElevi = file of Elev;  
   var FE : FisierElevi;  
       E : Elev;
```

Structura datelor în studiu este prezentată în *figura 1.8*. Subliniem faptul că elementul *EOF*, care indică sfîrșitul secvenței, nu este o componentă a fișierului.

Variabilele FN, FC, FE ș.a. de tip *fișier* se numesc fișiere logice, **fișiere PASCAL** sau, pur și simplu, **fișiere**. Spre deosebire de celelalte tipuri de date, valorile cărora se păstrează în memoria internă a calculatorului, datele fișierelor PASCAL se păstrează pe suporturile de informație ale echipamentelor periferice (discuri și benzi magnetice, discuri optice, hîrtia imprimantei sau a dispozitivului de citit

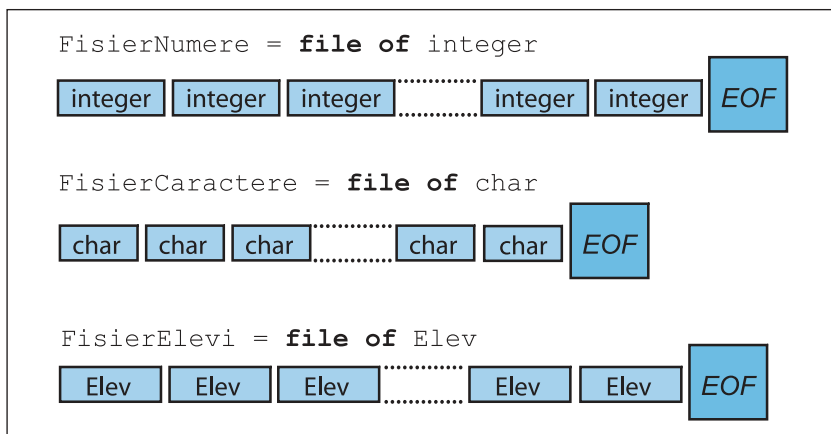


Fig. 1.8. Structura datelor de tip *FisierNumere*, *FisierCaractere* și *FisierElevi*

documente ș.a.). Informația pe suporturile în studiu este organizată în formă de **fișiere externe** în conformitate cu cerințele sistemului de operare. Prin urmare, înainte de a fi utilizată, o variabilă fișier trebuie asociată cu un fișier extern. Metodele de asociere sînt specifice implementării limbajului și sistemului de operare al calculatorului-gazdă.

În limbajul-standard asocierea se realizează prin includerea variabilelor de tip *fișier* ca argumente în antetul de program.

În Turbo PASCAL asocierea unei variabile de tip *fișier* f cu un fișier extern se realizează prin apelul de procedură

```
assign( $f$ ,  $s$ );
```

unde s este o expresie de tip **string** care specifică calea de acces și numele fișierului extern.

Exemple:

1) `assign(FN, 'A:\REZULTAT\R.DAT')`

– fișierul FN se asociază cu fișierul extern R.DAT din directorul REZULTAT de pe discul A.

2) `assign(FC, 'C:\A.CHR')`

– fișierul FC se asociază cu fișierul A.CHR din directorul-rădăcină al discului C.

3) `write('Dati un nume de fișier:');
readln(str);
assign(FE, str);`

– fișierul FE se asociază cu un fișier extern numele căruia este citit de la tastatură și depus în variabila de tip **string** str .

După executarea instrucțiunii `assign( $f$ ,  $s$ )` toate operațiile referitoare la fișierul PASCAL f se vor efectua asupra fișierului extern s .

Cele mai uzuale operații asupra unui fișier sînt citirea și scrierea unei componente.

Citirea unei componente se realizează printr-un apel de forma

```
read( f, v),
```

unde v este o variabilă declarată cu tipul de bază al fișierului f .

Scrierea unei componente se realizează printr-un apel de forma:

```
write( f, e),
```

unde e este o expresie asociată cu tipul de bază al fișierului f .

Exemple:

1) read(FN, n);

2) write(FC, c);

3) read(FE, E).

După **tipul operațiilor permise** asupra componentelor, fișierele se clasifică în:

- fișiere de intrare (este permisă numai citirea);
- fișiere de ieșire (este permisă numai scrierea);
- fișiere de actualizare (sînt permise atît scrierea, cît și citirea).

După **modul de acces** la componente, fișierele se clasifică în:

- fișiere cu acces secvențial sau secvențiale (accesul la componenta i este permis după ce s-a citit/scriș componenta $i - 1$);
- fișiere cu acces aleatoriu sau direct (orice componentă se poate referi direct prin numărul ei de ordine i în fișier).

Menționăm că în limbajul-standard sînt permise numai fișiere secvențiale de intrare sau ieșire.

Tipul fișierului (de intrare, ieșire sau de actualizare) și modul de acces (secvențial sau direct) se stabilesc printr-o operație de validare, numită **deschidere a fișierului**. Pentru aceasta, în limbajul-standard se utilizează următoarele proceduri:

reset(f) — pregătește un fișier existent pentru citire;

rewrite(f) — creează un fișier vid și îl pregătește pentru scriere.

Cînd prelucrarea componentelor se termină, fișierul trebuie închis. La **închiderea** unui fișier sistemul de operare înscrie elementul *EOF*, înregistrează fișierul extern nou-creat în directorul respectiv ș.a.m.d.

În limbajul-standard se consideră că fișierele vor fi închise implicit la terminarea execuției programului respectiv. În Turbo PASCAL fișierul f se închide prin apelul de procedură close(f).

În concluzie, prezentăm ordinea în care trebuie apelate procedurile destinate prelucrării datelor de tip *fișier*:

1) assign(f , s) — asocierea fișierului PASCAL f cu fișierul extern s ;

2) reset(f) / rewrite(f) — deschiderea fișierului f pentru citire/scriere;

3) read(f , v) / write(f , e) — citirea/scrierea unei componente a fișierului f ;

4) close(f) — închiderea fișierului f .

După închiderea fișierului, variabila f poate fi asociată cu un alt fișier extern.

Întrucât valorile variabilelor de tip fișier se păstrează pe suporturile externe de informație, în PASCAL **atribuirile de fișiere sînt interzise**.

Întrebări și exerciții

- ❶ Explicați termenii *fișier PASCAL*, *fișier extern*.
- ❷ Unde se păstrează datele unui fișier PASCAL? Care este destinația procedurii `assign`?
- ❸ Reprezentați pe un desen structura următoarelor tipuri de date:
 - a)

```
type Tabel = array [1..5, 1..10] of real;  
FisierTabele = file of Tabel;
```
 - b)

```
type Multime = set of 'A'..'C';  
FisierMultimi = file of Multime;
```
 - c)

```
type Punct = record x, y:real end;  
Segment = record A, B:Punct end;  
FisierSegmente = file of Segment;
```
- ❹ Pentru ce sînt necesare operațiile de deschidere și închidere a fișierelor? Cum se realizează aceste operații?
- ❺ Explicați destinația procedurilor `read` și `write`. Ce tip trebuie să aibă variabila v într-un apel de forma `read (f, v)`? Ce tip trebuie să aibă expresia e într-un apel de forma `write (f, e)`?
- ❻ Cum se clasifică fișierele în funcție de operațiile permise și modul de acces?
- ❼ Variabilele A și B sînt introduse prin declarația

```
var A, B : file of integer;
```

Este oare corectă instrucțiunea

```
A:=B
```

Argumentați răspunsul.

1.7. Fișiere secvențiale

Fie definițiile PASCAL

```
type FT = file of T;  
var f : FT; v : T;
```

prin care au fost introduse tipul *fișier* FT cu tipul de bază T , variabila de tip *fișier* f și variabila v de tipul T .

Pentru a deschide un **fișier secvențial de ieșire**, se apelează procedura `rewrite(f)`. În continuare în fișier se înscriu componentele respective. O componentă se înscrie printr-un apel de forma

```
write(f, e),
```

unde e este o expresie de tipul T . O instrucțiune de forma

```
write(f, e1, e2, ..., en)
```

este echivalentă cu secvența de instrucțiuni

```
write(f, e1) ; write(f, e2); ...; write(f, en).
```

După înscrierea ultimei componente fișierul trebuie închis.

Exemplu:

```
Program P89;
{ Crearea unui fișier cu componente de tipul Elev }
type Elev = record
    Nume : string;
    Prenume : string;
    NotaMedie : real;
end;
    FisierElevi = file of Elev;
var FE : FisierElevi;
    E : Elev;
    str : string;
    i, n : integer;
begin
    write('Dati numele fișierului de creat: ');
    readln(str);

    assign(FE, str);      { asociaza FE cu numele din str }
    rewrite(FE);          { deschide FE pentru scriere }

    write('Dati numarul de elevi: '); readln(n);

    for i:=1 to n do
        begin
            writeln('Dati datele elevului ', i);

            { citește cimpurile variabilei E de la tastatura }
            write('Numele: ');    readln(E.Nume);
            write('Prenumele: '); readln(E.Prenume);
            write('Nota medie: '); readln(E.NotaMedie);

            { scrie valoarea variabilei E în fișierul FE }
```

```

        write(FE, E);
    end;
    close(FE);          { inchide fisierul FE }
    readln;
end.

```

Pentru a **deschide un fișier secvențial de intrare**, se apelează procedura `reset(f)`. Componenta curentă se citește din fișier printr-un apel de forma:

```
read(f, v).
```

O instrucțiune de forma

```
read(f, v1, v2, ..., vn)
```

este echivalentă cu secvența de instrucțiuni

```
read(f, v1); read(f, v2); ..., read(f, vn).
```

Sfârșitul de fișier este semnalizat de funcția booleană `eof(f)` care ia valoarea `true` după citirea ultimei componente.

Exemplu:

```

Program P90;
{ Citirea unui fisier cu componente de tipul Elev }
type Elev = record
    Nume : string;
    Prenume : string;
    NotaMedie : real;
end;
    FisierElevi = file of Elev;
var FE : FisierElevi;
    E : Elev;
    str : string;
begin
    write('Dati numele fisierului de citit: ');
    readln(str);

    assign(FE, str); { asociaza FE cu numele din str }
    reset(FE);       { deschide fisierul FE pentru citire }

    while not eof(FE) do
        begin
            { citeste E din fisierul FE }
            read(FE, E);
            { afiseaza E pe ecran }
            writeln(E.Nume, ' ', E.Prenume, ': ',
                    E.NotaMedie : 5:2);
        end;

```

```
close(FE);           { inchide fisierul FE }
readln;
end.
```

Subliniem faptul că numărul de componente ale unui fișier nu este cunoscut din declarația tipului respectiv. Teoretic, într-un fișier secvențial de ieșire poate fi înscris un număr infinit de componente. Practic, numărul componentelor este limitat de capacitatea de memorare a suportului extern de informație. Citirea consecutivă a componentelor unui fișier secvențial de intrare se încheie când se ajunge la elementul *EOF*.

Întrebări și exerciții

- ❶ Cîte componente poate avea un fișier? În ce ordine se scriu/citesc componentele unui fișier secvențial?
- ❷ Se consideră următoarele tipuri de date:

```
type Data = record
    Ziua : 1..31;
    Luna : 1..12;
    Anul : integer;
end;
Persoana = record
    NumePrenume : string;
    DataNasterii : Data;
end;
FisierPersoane = file of Persoana;
```

Elaborați un program care citește de la tastatură datele referitoare la n persoane și le înregistrează într-un fișier. Creați fișierele `FILE1.PRS`, `FILE2.PRS`, `FILE3.PRS` care trebuie să conțină datele referitoare, respectiv, la 2, 7 și 10 persoane.

- ❸ Elaborați un program care citește fișiere create de programul din exercițiul precedent și afișează pe ecran:
 - a) toate persoanele din fișier;
 - b) persoanele născute în anul a ;
 - c) persoanele născute pe data $z.l.a$;
 - d) persoana cea mai în vârstă;
 - e) persoana cea mai tânără.
- ❹ Elaborați un program care afișează pe ecran media aritmetică a numerelor înscrise într-un fișier de tipul **file of real**.
- ❺ Într-un fișier de tipul **file of char** sînt înscrise caractere arbitrare. Elaborați un program care afișează pe ecran numărul vocalelor din fișier.
- ❻ Comentați următorul program:

```
Program P91;
{ Eroare }
type FisierNumere = file of integer;
```

```

var FN : FisierNumere;
    i : integer;
    r : real;
    s : string;
begin
    writeln('Dati numele fisierului de creat: ');
    readln(s);
    assign(FN, s);
    rewrite(FN);
    i:=1;
    write(FN, i);
    i:=10;
    write(FN, i);
    r:=20;
    write(FN, r);
    close(FN);
end.

```

1.8. Fișiere text

E cunoscut faptul că datele fișierelor PASCAL se păstrează pe suporturile externe de informație. În cazul fișierelor definite prin declarații de forma **file of T** componentele de tip *T* sînt reprezentate pe suporturile respective în **forma internă**, și anume, prin secvențe de cifre binare. Acest mod de reprezentare a datelor este convenabil în cazul memoriilor externe (discurile și benzile magnetice, discurile optice ș.a.). În cazul echipamentelor de intrare–ieșire (tastatura, ecranul, imprimanta etc.) datele respective trebuie reprezentate în **forma externă**, și anume, prin secvențe de caractere.

Pentru a facilita interacțiunea între om și sistemul de calcul, în PASCAL informația destinată utilizatorului se reprezintă în formă de fișiere *text*. Un fișier *text* este format dintr-o secvență de caractere divizată în linii (fig. 1.9). Lungimea liniilor este variabilă. Sfîrșitul fiecărei linii este indicat de un element special, notat *EOL* (*End Of Line* – sfîrșit de linie). Întrucît lungimea liniilor este

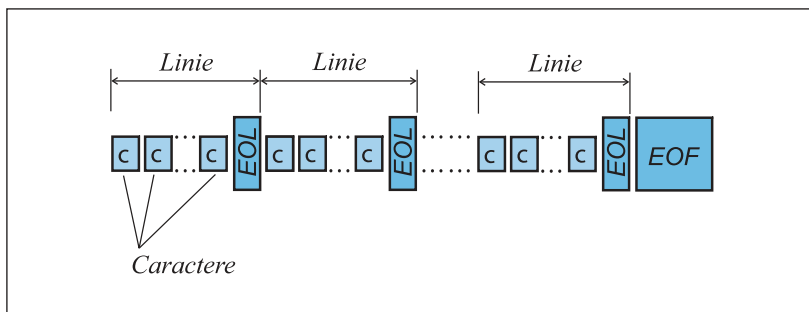


Fig. 1.9. Structura unui fișier text

variabilă, poziția unei linii în cadrul fișierului nu poate fi calculată din timp. În consecință, accesul la componentele fișierelor *text* este **secvențial**.

Un fișier *text* se definește printr-o declarație de forma:

```
var f : text;
```

tipul predefinit `text` fiind cunoscut oricărui program PASCAL. Subliniem faptul că tipurile `text` și `file of char` sînt distincte, întrucît fișierul `file of char` nu include elemente *EOL* (fig. 1.8).

Fișierele *text* pot fi prelucrate cu ajutorul procedurilor cunoscute, aplicabile oricărui tip de fișiere: `assign`, `reset`, `rewrite`, `read`, `write`, `close`. În completare, limbajul include proceduri speciale, destinate prelucrării elementelor *EOL*:

`writeln(f)` – înscrie în fișier elementul *EOL* (sfîrșit de linie);

`readln(f)` – trece la linia următoare.

Sfîrșitul de linie este semnalat de funcția booleană `eofln(f)` care ia valoarea `true` după citirea ultimului caracter din linie.

O instrucțiune de forma

```
writeln(f, e1, e2, ..., en)
```

este echivalentă cu secvența de instrucțiuni

```
write(f, e1, e2, ..., en); writeln(f).
```

O instrucțiune de forma

```
readln(f, v1, v2, ..., vn)
```

este echivalentă cu secvența de instrucțiuni

```
read(f, v1, v2, ..., vn); readln(f).
```

Pentru introducerea și extragerea datelor, de regulă, se utilizează fișierele *text* predefinite `Input` și `Output`. Fișierul `Input` este destinat numai pentru operații de citire și este asociat cu fișierul de intrare al sistemului de operare (de regulă, tastatura). Fișierul `Output` este destinat numai pentru operații de scriere și este asociat cu fișierul-standard de ieșire al sistemului de operare (de regulă ecranul). Aceste fișiere sînt deschise și închise automat la începutul și, respectiv, la sfîrșitul execuției programului. Dacă numele fișierului nu este specificat în lista parametrilor unui apel de subalgoritm, se presupune că fișierul *text* implicit este `Input` sau `Output`, în funcție de natura subalgoritmului. De exemplu, `read(c)` este echivalent cu `read(Input, c)`, iar `write(c)` este echivalent cu `write(Output, c)`.

Pentru exemplificare prezentăm programul P93 care creează pe discul curent fișierul `text FILE.TXT`. Liniile fișierului sînt introduse de la tastatură (fișierul `Input`). Sfîrșitul liniei se indică prin acționarea tastei `<ENTER>`, iar sfîrșitul fișierului prin acționarea tastelor `<CTRL+Z>`, `<ENTER>`.

Program P93;

```
{ Crearea fisierului text FILE.TXT }
var F : text;
    c : char;
begin
  assign(F, 'FILE.TXT'); { asociaza F cu FILE.TXT }
  rewrite(F);             { deschide fisierul F pentru scriere }
  while not eof do        { eof(Input) }
    begin
      while not eoln do   { eoln(Input) }
        begin
          read(c);         { citește c din Input }
          write(F, c);     { scrie c în F }
        end;
      writeln(F);          { scrie EOL în F }
      readln;              { trece la linia următoare din Input }
    end;
  close(F);               { închide F }
end.
```

Programul ce urmează afișează conținutul fișierului `FILE.TXT` pe ecran.

Program P94;

```
{ Citirea fisierului text FILE.TXT }
var F : text;
    c : char;
begin
  assign(F, 'FILE.TXT'); { asociaza F cu FILE.TXT }
  reset(F);              { deschide F pentru citire }
  while not eof(F) do
    begin
      while not eoln(F) do
        begin
          read(F, c);     { citește c din F }
          write(c);       { scrie c în Output }
        end;
      readln(F);          { trece la linia următoare din F }
      writeln;            { scrie EOL în Output }
    end;
  close(F);              { închide F }
  readln;
end.
```

Explorarea caracter cu caracter a fișierelor *text* este greoaie în situația când secvențele de caractere din text trebuie interpretate ca formând date de tip *integer*, *real*, *boolean*, *șir de caractere*. Conversiunea între forma externă

și reprezentarea internă a acestor tipuri ar cădea în sarcina programatorului. În consecință, procedurile de citire/înscrisiere sînt extinse în felul următor.

În cazul fișierelor *text* variabila v dintr-un apel `read( $f$ ,  $v$ )` poate fi de tipul `integer`, `real`, `char` sau *șir de caractere*. La citire secvența de caractere care reprezintă variabila v va fi transformată în reprezentarea internă.

Expresia e dintr-un apel `write( $f$ ,  $e$ )` poate fi urmată de specificatori de format. Valoarea expresiei poate fi de tipul `integer`, `real`, `boolean`, `char` sau *șir de caractere*. La scriere valoarea respectivă este transformată din reprezentarea internă într-o secvență de caractere.

Secvențele de caractere citite/scrise de procedurile `read/write` se conformează sintaxei constantelor de tipul variabilei/expresiei v/e .

Pentru exemplificare prezentăm programul P95 care citește de la tastatură cîte trei numere reale a , b , c pe care le scrie în fișierul `IN.TXT`. Apoi citind aceste trei numere, reprezentînd laturile unui triunghi, scrie în fișierul `OUT.TXT` numerele a , b și c , semiperimetrul p și aria triunghiului s . În continuare, conținutul fișierului `OUT.TXT` este afișat pe ecran.

Program P95;

```
{ Prelucrarea fisierelor IN.TXT si OUT.TXT }
var F, G : text;
    a, b, c, p, s : real;
    str : string;
begin
    assign(F, 'IN.TXT'); { asociaza F cu IN.TXT }
    rewrite(F);          { deschide F pentru scriere }

    writeln('Dati numerele reale a, b, c:');
    while not eof do
        begin
            readln(a, b, c); { citeste a, b, c de la tastatura }
            writeln(F, a:8:2, b:8:2, c:8:2); { scrie a, b, c, in F }
        end;
    close(F);              { inchide F }

    reset(F);              { deschide F pentru citire }
    assign(G, 'OUT.TXT'); { asociaza G cu OUT.TXT }
    rewrite(G);            { deschide G pentru scriere }

    while not eof(F) do
        begin
            readln(F, a, b, c); { citeste a, b, c din F }
            write(G, a:8:2, b:8:2, c:8:2); { scrie a, b, c in G }
            p:=(a+b+c)/2;
            s:=sqrt(p*(p-a)*(p-b)*(p-c));
```

```

        writeln(G, p:15:2, s:15:4); { scrie p, s in G }
    end;

    close(F);           { inchide F }
    close(G);           { inchide G }

    reset(G);           { deschide G pentru citire }
    while not eof(G) do
        begin
            readln(G, str); { citește str din G }
            writeln(str);   { afișează str pe ecran }
        end;
    close(G);           { inchide G }
    readln;
end.

```

Pentru datele de intrare

```

1 1 1 <ENTER>
3 4 6 <ENTER>
<CTRL+Z ><ENTER>

```

programul P95 afișează pe ecran:

```

1.00 1.00 1.00 1.50 0.4330
3.00 4.00 6.00 6.50 5.3327

```

Întrebări și exerciții

- ❶ Care este diferența dintre un fișier *text* și un fișier **file of char**?
- ❷ Explicați semnificația elementelor *EOL* și *EOF*.
- ❸ Care este diferența dintre procedurile **read** și **readln**? Dar dintre procedurile **write** și **writeln**?
- ❹ Lansați în execuție următorul program:

```

Program P96;
{ Asocierea fisierului FN cu consola }
type FisierNumere = file of integer;
var FN : FisierNumere;
    i : integer;
begin
    assign(FN, 'CON');
    rewrite(FN);
    i:=1;
    write(FN, i);
    i:=2;
    write(FN, i);

```

```
i:=3;  
write(FN, i);  
close(FN);  
readln;  
end.
```

Explicați rezultatele afișate pe ecran.

- ⑤ Elaborați un program care afișează pe ecran conținutul oricărui fișier *text*.
- ⑥ Elaborați un program care afișează pe ecran numărul de vocale dintr-un fișier *text*.
- ⑦ Datele de intrare ale unui program sînt înmagazinate într-un fișier *text*. Fiecare linie a fișierului conține două numere întregi și trei numere reale separate prin spații. Elaborați un program care afișează suma numerelor întregi și suma numerelor reale din fiecare linie pe ecran.
- ⑧ Datele de intrare ale unui program sînt înmagazinate într-un fișier *text*. Fiecare linie a fișierului conține trei numere reale separate prin spațiu și unul din cuvintele ADMIS, RESPINS. Elaborați un program care:
 - a) afișează conținutul fișierului în studiu pe ecran;
 - b) creează o copie de rezervă a fișierului;
 - c) creează un fișier *text* liniile căruia conțin media celor trei numere reale din liniile respective ale fișierului de intrare;
 - d) afișează pe ecran liniile fișierului de intrare, precedate de numerele de ordine 1, 2, 3 ș.a.m.d.
- ⑨ Fiecare linie a unui fișier *text* conține următoarele date, separate prin spații:
 - numărul de ordine (**integer**);
 - numele (un **string** ce nu conține spații);
 - prenumele (un **string** ce nu conține spații);
 - nota la disciplina 1 (**real**);
 - nota la disciplina 2 (**real**);
 - nota la disciplina 3 (**real**).Elaborați un program care:
 - a) creează o copie de rezervă a fișierului în studiu;
 - b) afișează conținutul fișierului pe ecran;
 - c) creează un fișier *text* liniile căruia conțin următoarele date separate prin spații:
 - numărul de ordine (**integer**);
 - numele (**string**);
 - prenumele (**string**);
 - nota medie (**real**).

Fișierul creat în punctul c trebuie afișat pe ecran.

Test de autoevaluare nr. 1

1. Se consideră declarațiile

```
type Obiect = (      Istoria, Geografia, Matematica,
                  Informatica, Fizica);
Nota = 1..10;
SituatiaScolara = array [Obiect] of Nota;
```

Reprezentați pe un desen structura datelor de tipul `SituatiaScolara`.

2. Precizați tipul indicilor și tipul componentelor tipului de dată `OrarulLectiilor` din declarațiile ce urmează:

```
type ZiDeScoala = (L, Ma, Mi, J, V, S);
Lectie = 1..6;
Obiect = (LimbaRomana, LimbaModerna, Istoria,
          Geografia, Matematica, Informatica, Fizica,
          Chimia);
OrarulLectiilor = array [ZiDeScoala, Lectie] of Obiect;
```

3. Se consideră declarațiile:

```
type Tablou = array [1..10] of integer;
var x, y : Tablou;
```

Scrieți expresia aritmetică a cărei valoare este:

- a) suma primelor patru componente ale variabilei `x`;
- b) suma ultimelor patru componente ale variabilei `y`;
- c) valoarea absolută a componentei a treia a variabilei `x`;
- d) valoarea absolută a componentei a șasea a variabilei `y`;
- e) suma primei componente a variabilei `x` și a ultimei componente a variabilei `y`.

4. Se consideră n ($n \leq 50$) numere întregi $a_1, a_2, a_3, \dots, a_n$. Elaborați un program PASCAL care citește numerele respective de la tastatură și le afișează pe ecran în ordinea inversă citirii: $a_n, \dots, a_3, a_2, a_1$.

5. Ce operații pot fi efectuate asupra șirurilor de caractere de tip `string`? Precizați tipul rezultatelor acestor operații.

6. Elaborați un program PASCAL care afișează pe ecran în ordine inversă șirul de caractere citit de la tastatură. De exemplu, șirul 'soare' va fi afișat la ecran ca 'eraos'.

7. Reprezențați pe un desen structura datelor de tipul `Data` și `Persoana` din declarațiile ce urmează:

```
type Data = record
    Ziua : 1..31;
    Luna : 1..12;
    Anul : integer;
end;
Persoana = record
    NumePrenume : string;
    DataNasterii : Data;
end;
```

8. Se consideră următoarele tipuri de date:

```
type Angajat = record
    NumePrenume : string;
    Salariu : real;
end;
ListaDePlata = array[1..100] of Angajat;
```

Elaborați un program care citește de la tastatură datele despre n ($n \leq 100$) angajați și afișează pe ecran informațiile despre angajatul (angajații) care au cel mai mare salariu.

9. Care este destinația instrucțiunii **with**?

10. Enumerați valorile posibile ale variabilelor din declarațiile ce urmează:

```
var V : set of 'X'..'Z';
    I : set of 8..9;
```

11. Se consideră șiruri de caractere formate din literele mari ale alfabetului latin. Elaborați un program care afișează pe ecran numărul de vocale din șirul de caractere S citit de la tastatură.

12. Unde se păstrează datele fișierelor PASCAL? Care este destinația procedurii **assign**?

13. Cum se clasifică fișierele în funcție de operațiile permise și modul de acces?

14. Se consideră următoarele tipuri de date:

```
type Angajat = record
    NumePrenume : string;
    Salariu : real;
end;
FisierAngajati = file of Angajat;
```

Elaborați un program care creează fișierul SALARII.DAT și înscrie în acest fișier datele referitoare la n angajați. Datele respective se citesc de la tastatură.

15. Elaborați un program PASCAL care afișează pe ecran conținutul fișierului SALARII.DAT creat în itemul precedent.

16. Datele de intrare ale unui program sînt înmagazinate în fișierul *text* REZULTAT.TXT. Fiecare linie a fișierului conține cîte două numere reale separate prin spațiu și unul din cuvintele BUN, DEFECT. Elaborați un program care:

a) creează fișierul *text* MEDIA.TXT liniile căruia conțin media celor două numere reale și cuvintele BUN sau DEFECT din liniile respective ale fișierului de intrare;

b) afișează pe ecran liniile fișierului creat.

2.1. Cantitatea de informație

Sensul uzual al cuvîntului **informație** „știre, comunicare verbală, scrisă sau transmisă prin alte metode despre anumite fapte, evenimente, activități etc.” se concretizează într-un compartiment special al matematicii, denumit **teoria informației**. Conform acestei teorii, **sursa de informație** se descrie printr-o variabilă S care poate lua valori dintr-o mulțime finită de elemente distincte $\{s_1, s_2, \dots, s_n\}$. Se consideră că valorile curente ale variabilei S nu sînt cunoscute din timp. E cunoscută numai mulțimea $\{s_1, s_2, \dots, s_n\}$, **denumită mulțimea mesajelor posibile**.

De exemplu, semaforul de circulație poate fi reprezentat ca o sursă de informație, mulțimea de mesaje posibile ale căruia este $\{\text{verde}, \text{galben}, \text{roșu}\}$. Aparatul de telegrafiat reprezintă o sursă de informație, mulțimea de mesaje posibile ale căruia include literele $A, B, C, \dots, Z$, cifrele $0, 1, 2, \dots, 9$ și semnele de punctuație. Mesajele posibile ale tastaturii sînt: „Este acționată tasta A ”, „Este acționată tasta B ”, ..., „Este acționată tasta $F1$ ”, „Este acționată tasta $F2$ ”, ..., „Sînt acționate concomitent tastele $CTRL$ și $BREAK$ ” etc.

Mesajele se transmit de la sursă către destinatar printr-un mediu fizic, numit **canal de transmisie** (fig. 2.1). De exemplu, mesajele telegrafice se transmit prin fir, mesajele radio prin eter, mesajele tastaturii printr-un set de conductori. **Perturbațiile** (zgomotele) din mediul fizic amintit pot altera mesajele transmise.

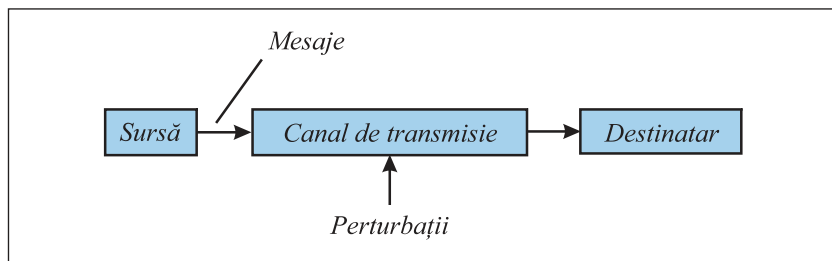


Fig. 2.1. Schema generală a unui sistem de transmisie a informației

Evident, valoarea curentă a variabilei S devine cunoscută destinatarului numai după recepționarea mesajului respectiv.

Cantitatea de informație I ce este conținută într-un mesaj emis de sursă se determină din relația:

$$I = \log_a n,$$

unde n este numărul de mesaje posibile ale sursei. Valoarea concretă a constantei a se stabilește prin alegerea **unității de măsură a cantității de informație**. De obicei, ca unitate de măsură se utilizează **bitul**.

Un bit este cantitatea de informație din mesajul unei surse cu numai două mesaje posibile.

Prin urmare, ca și în cazul altor mărimi (lungimea, masa, temperatura etc.), cantitatea de informație se măsoară prin compararea cu **etalonul**. Întrucât pentru sursa-etalon $n = 2$, din ecuația:

$$\log_a 2 = 1 \text{ (bit)}$$

obținem $a = 2$. În consecință, cantitatea de informație I , măsurată în biți, se determină din relația:

$$I = \log_2 n \text{ (bit)}.$$

În *tabelul 2.1* sînt prezentate valorile frecvent utilizate ale funcției $\log_2 n$.

Tabelul 2.1

Valorile funcției $\log_2 n$

n	$\log_2 n$	n	$\log_2 n$
1	0,000	21	4,392
2	1,000	22	4,459
3	1,585	23	4,524
4	2,000	24	4,585
5	2,322	25	4,644
6	2,585	26	4,700
7	2,807	27	4,755
8	3,000	28	4,807
9	3,170	29	4,858
10	3,322	30	4,907
11	3,459	31	4,954
12	3,585	32	5,000
13	3,700	33	5,044
14	3,807	34	5,087
15	3,907	35	5,129
16	4,000	36	5,170
17	4,087	37	5,209
18	4,170	38	5,248
19	4,248	39	5,285
20	4,322	40	5,322

Să analizăm câteva exemple. Cantitatea de informație a unui mesaj de semafor este de

$$I = \log_2 3 \approx 1,585 \text{ biți.}$$

Cantitatea de informație a unei litere a alfabetului latin $\{A, B, C, \dots, Z\}$, $n = 26$, este de

$$I = \log_2 26 \approx 4,700 \text{ biți.}$$

Cantitatea de informație a unei litere a alfabetului grec $\{A, B, \Gamma, \Delta, \dots, \Omega\}$, $n = 24$, este de

$$I = \log_2 24 \approx 4,585 \text{ biți.}$$

Dacă se cunoaște cantitatea de informație I ce este conținută într-un mesaj, cantitatea totală de **informație emisă** de sursă se determină din relația:

$$V = N I,$$

unde N este numărul de mesaje transmise.

Cantitățile mari de informație se exprimă prin multiplii unui bit:

$$1 \text{ Kilobit (Kbit)} = 2^{10} = 1\,024 \text{ biți } (\approx 10^3 \text{ biți});$$

$$1 \text{ Megabit (Mbit)} = 2^{20} = 1\,048\,576 \text{ biți } (\approx 10^6 \text{ biți});$$

$$1 \text{ Gigabit (Gbit)} = 2^{30} \approx 10^9 \text{ biți};$$

$$1 \text{ Terabit (Tbit)} = 2^{40} \approx 10^{12} \text{ biți};$$

$$1 \text{ Petabit (Pbit)} = 2^{50} \approx 10^{15} \text{ biți.}$$

Întrebări și exerciții

- ❶ Cum se definește o sursă de informație? Dați câteva exemple.
- ❷ Care este destinația canalului de transmisie?
- ❸ Cum se determină cantitatea de informație dintr-un mesaj? Dar din N mesaje?
- ❹ Care este unitatea de măsură a informației și ce semnificație are ea?
- ❺ Determinați cantitatea de informație într-un mesaj al surselor cu următoarele mesaje posibile:
 - a) literele mari și mici ale alfabetului latin;
 - b) literele mari și mici ale alfabetului grec;
 - c) literele mari și mici ale alfabetului român;
 - d) cifrele zecimale 0, 1, 2, ..., 9;
 - e) cifrele 0, 1, 2, ..., 9, semnele +, −, ×, / și parantezele ();
 - f) indicațiile numerice de forma $hh:mm$ (hh – ora, mm – minutele) ale unui ceas electronic;
 - g) indicațiile numerice de forma $hh:mm:ss$ (ss – secunde) ale unui ceas electronic;
 - h) indicațiile numerice de forma $zz.ll.aa$ (zz – ziua, ll – luna, aa – anul) ale unui calendar electronic.
- ❻ Pentru fiecare dintre sursele indicate în exercițiul 5 determinați cantitatea de informație ce este conținută în 1 000 de mesaje emise de sursă.
- ❼ Elaborați un program care calculează cantitatea de informație din N mesaje emise de o sursă cu n mesaje posibile.

2.2. Codificarea și decodificarea informației

Se numește **semn** un element al unei mulțimi finite de obiecte ce se pot distinge. O mulțime de semne ordonate liniar se numește **alfabet**.

Prezentăm în continuare unele dintre nenumăratele alfabetele folosite de oameni:

- a) alfabetul cifrelor zecimale: 0, 1, 2, ..., 9;
- b) alfabetul literelor latine mari: A, B, C, ..., Z;
- c) mulțimea semnelor zodiacului;
- d) mulțimea fazelor lunii.

O importanță deosebită o au alfabetele de numai două semne. Aceste alfabetele se numesc **alfabete binare**, iar semnele respective – **semne binare**.

Prezentăm câteva exemple de alfabetele binare:

- a) cifrele {0, 1};
- b) perechea de culori {roșu, galben};
- c) perechea de stări {închis, deschis};
- d) perechea de răspunsuri {da, nu};
- e) perechea de tensiuni {0V, 2V};
- f) perechea de stări {magnetizat, nemagnetizat};
- g) perechea de semne {+, -} etc.

S-a convenit ca semnele unui alfabet binar să fie reprezentate prin cifrele {0, 1}, denumite **cifre binare** (*binary digit*).

Un șir finit din m semne, dintre care unele se pot repeta, formează un **cuvânt**, m reprezentând **lungimea cuvântului**. Cuvintele formate din semne binare se numesc **cuvinte binare**. Evident, cuvintele pot avea lungime variabilă sau constantă. În ultimul caz ele se numesc **cuvinte m -poziționale**. În continuare prezentăm unele mulțimi de cuvinte cu lungime constantă:

- 1-poziționale: {0, 1};
- 2-poziționale: {00, 01, 10, 11};
- 3-poziționale: {000, 001, 010, 011, 100, 101, 110, 111};
- 4-poziționale: {0000, 0001, ..., 1110, 1111}.

Se observă că cuvintele $(m + 1)$ -poziționale se formează câte două din cuvintele m -poziționale prin adăugarea cifrelor binare 0 și 1. Prin urmare, mulțimea cuvintelor m -poziționale include 2^m cuvinte distincte.

Cuvintele binare se utilizează pentru reprezentarea, transmiterea, păstrarea și prelucrarea mesajelor $s_1, s_2, \dots, s_n$ ale sursei de informație (fig. 2.2).

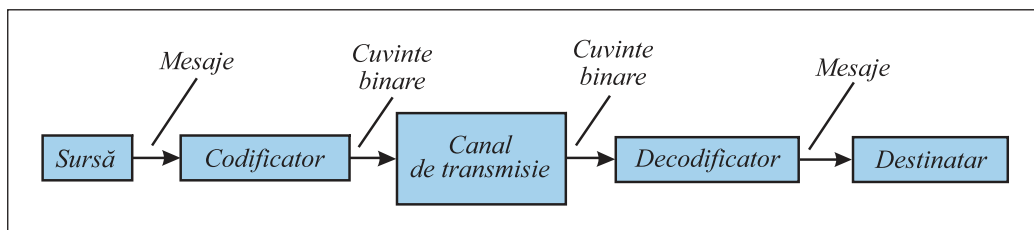


Fig. 2.2. Codificarea și decodificarea mesajelor în sistemele de transmisie a informației

Regula de transformare a mesajelor în cuvinte se numește cod, iar operația respectivă – codificare. Operația inversă codificării se numește decodificare. Dispozitivele tehnice care realizează operațiile în cauză se numesc, respectiv, codificator și decodificator.

Cel mai simplu este codul în care mesajelor posibile $s_1, s_2, \dots, s_n$ le corespund cuvinte binare de lungime constantă m . Acest cod denumit **cod m -pozițional** poate fi definit cu ajutorul unui tabel. În figura 2.3 sînt prezentate tabelele respective pentru surse cu $n = 2, 3, 4, \dots, 8$ mesaje posibile.

$n=2, m=1$		$n=3, m=2$		$n=4, m=2$	
s_1	0	s_1	00	s_1	00
s_2	1	s_2	01	s_2	01
		s_3	10	s_3	10
				s_4	11

$n=5, m=3$		$n=6, m=3$		$n=7, m=3$		$n=8, m=3$	
s_1	000	s_1	000	s_1	000	s_1	000
s_2	001	s_2	001	s_2	001	s_2	001
s_3	010	s_3	010	s_3	010	s_3	010
s_4	011	s_4	011	s_4	011	s_4	011
s_5	100	s_5	100	s_5	100	s_5	100
		s_6	101	s_6	101	s_6	101
				s_7	110	s_7	110
						s_8	111

Fig. 2.3. Coduri de cuvinte cu lungime constantă (coduri m -poziționale)

Operațiile de codificare și decodificare constau în extragerea datelor necesare din tabel. Evident, decodificarea va fi univocă numai atunci cînd cuvintele binare incluse în tabel sînt distincte. Acest lucru este posibil dacă lungimea m a cuvintelor de cod satisface inegalitatea

$$2^m \geq n.$$

După logaritmare obținem:

$$m \geq \log_2 n.$$

Întrucît expresia $\log_2 n$ exprimă cantitatea de informație, se poate afirma:

Lungimea cuvintelor unui cod pozițional trebuie să fie mai mare sau egală cu cantitatea de informație a unui mesaj.

De exemplu, lungimea cuvintelor pentru codificarea literelor mari ale alfabetului latin $\{A, B, C, \dots, Z\}$, $n = 26$, se determină din relația

$$m \geq \log_2 26 \approx 4,700.$$

Stabilind $m = 5$, putem forma cuvintele binare ale codului 5-pozițional:

$A - 00000$

$B - 00001$

$C - 00010$

$D - 00011$

$E - 00100$

...

$Z - 11001.$

Un astfel de cod a fost propus de filosoful și omul de stat englez Francis Bacon încă în anul 1580.

Algoritmii de elaborare a **codurilor cu cuvinte de lungime variabilă** sînt mult mai complicați și se studiază în cursurile avansate de informatică.

Întrebări și exerciții

- ❶ Ce este un alfabet? Dați exemple de alfabet binare.
- ❷ Cum se reprezintă semnele oricărui alfabet binar?
- ❸ Explicați cum pot fi formate cuvintele binare $(m+1)$ -poziționale. Care este numărul cuvintelor binare m -poziționale distincte?
- ❹ Care este destinația unui cod? Cum se definește codul m -pozițional?
- ❺ Cum se efectuează codificarea și decodificarea mesajelor în cazurile cînd codul este definit printr-un tabel?
- ❻ Codificați mesajele s_3, s_4 și s_6 ale unei surse cu 7 mesaje posibile. Utilizați codul 3-pozițional din *figura 2.3*.
- ❼ Decodificați mesajele 100, 000 și 010, reprezentate în codul 3-pozițional din *figura 2.3*, $n = 5$.
- ❽ Cum se determină numărul de semne binare necesare pentru formarea cuvintelor de lungime constantă ale unui cod?
- ❾ Utilizînd codul 3-pozițional din *figura 2.3*, $n = 6$, codificați șirul de mesaje $s_1, s_2, s_6, s_5, s_3, s_6, s_3, s_2, s_1$.
- ❿ Cum influențează cantitatea de informație a unui mesaj asupra lungimii cuvintelor de cod?
- ⓫ Explicați sensul termenilor **cantitate de informație** și **informație**.
- ⓬ Elaborați un program care codifică și decodifică literele alfabetului latin. Se va utiliza codul propus de Francis Bacon.
- ⓭ Elaborați un program care alcătuiește tabelul unui cod m -pozițional pentru o sursă cu n mesaje posibile.

2.3. Coduri frecvent utilizate

Orice cod, utilizat pentru reprezentarea, transmiterea, păstrarea și prelucrarea informației, trebuie să fie econom și insensibil la perturbații, iar echipamentele respective de codificare și decodificare – să fie simple. Pe parcursul dezvoltării tehnicii de calcul au fost elaborate mai multe coduri. Aceste coduri se clasifică în coduri numerice și coduri alfanumerice.

Codurile numerice oferă posibilitatea reprezentării cifrelor $\{0, 1, 2, \dots, 9\}$ prin cuvinte binare 4-poziționale. Exemple de coduri numerice sînt prezentate în *tabelul 2.2*.

Coduri numerice

<i>Cifra</i>	<i>Denumirea codului</i>			
	<i>Direct</i>	<i>Gray</i>	<i>Aiken</i>	<i>Exces 3</i>
0	0000	0000	0000	0011
1	0001	0001	0001	0100
2	0010	0011	0010	0101
3	0011	0010	0011	0110
4	0100	0110	0100	0111
5	0101	0111	1011	1000
6	0110	0101	1100	1001
7	0111	0100	1101	1010
8	1000	1100	1110	1011
9	1001	1101	1111	1100

Codurile alfanumerice reprezintă prin cuvinte binare cifrele 0, 1, 2, ..., 9, literele mari și mici ale alfabetului, semnele de punctuație, semnele operațiilor aritmetice etc. În *tabelul 2.3* este prezentat codul **ASCII** (*American Standard Code for Information Interchange*), inventat în anul 1968.

Tabelul 2.3

Codul ASCII

<i>Simbol</i>	<i>Cuvînt binar</i>	<i>Echivalent zecimal</i>	<i>Simbol</i>	<i>Cuvînt binar</i>	<i>Echivalent zecimal</i>
Spațiu	0100000	32	P	1010000	80
!	0100001	33	Q	1010001	81
"	0100010	34	R	1010010	82
#	0100011	35	S	1010011	83
\$	0100100	36	T	1010100	84
%	0100101	37	U	1010101	85
&	0100110	38	V	1010110	86
'	0100111	39	W	1010111	87
(	0101000	40	X	1011000	88
)	0101001	41	Y	1011001	89
*	0101010	42	Z	1011010	90
+	0101011	43	[	1011011	91
,	0101100	44	\	1011100	92
-	0101101	45	]	1011101	93
.	0101110	46	^	1011110	94
/	0101111	47	_	1011111	95

<i>Simbol</i>	<i>Cuvînt binar</i>	<i>Echivalent zecimal</i>	<i>Simbol</i>	<i>Cuvînt binar</i>	<i>Echivalent zecimal</i>
0	0110000	48	`	1100000	96
1	0110001	49	a	1100001	97
2	0110010	50	b	1100010	98
3	0110011	51	c	1100011	99
4	0110100	52	d	1100101	100
5	0110101	53	e	1100101	101
6	0110110	54	f	1100110	102
7	0110111	55	g	1100111	103
8	0111000	56	h	1101000	104
9	0111001	57	i	1101001	105
:	0111010	58	j	1101010	106
;	0111011	59	k	1101011	107
<	0111100	60	l	1101100	108
=	0111101	61	m	1101101	109
>	0111110	62	n	1101110	110
?	0111111	63	o	1101111	111
@	1000000	64	p	1110000	112
A	1000001	65	q	1110001	113
B	1000010	66	r	1110010	114
C	1000011	67	s	1110011	115
D	1000100	68	t	1110100	116
E	1000101	69	u	1110101	117
F	1000110	70	v	1110110	118
G	1000111	71	w	1110111	119
H	1001000	72	x	1111000	120
I	1001001	73	y	1111001	121
J	1001010	74	z	1111010	122
K	1001011	75	{	1111011	123
L	1001100	76		1111100	124
M	1001101	77	}	1111101	125
N	1001110	78	~	1111110	126
O	1001111	79	Del	1111111	127

Acest cod este 7-pozițional și include $2^7 = 128$ de simboluri. Primele 32 de simboluri (cuvintele binare 0000000, 0000001, 0000010, ..., 0011111) specifică detaliile tehnice ale transmisiunilor de informații și nu au fost incluse în tabel. Cuvintele binare 0100000, 0100001, 0100010, ..., 1111110 reprezintă caracterele impri-

mabile din textele în limba engleză. Cuvîntul 1111111 reprezintă caracterul neimprimabil *Delete* (Anulare).

Codificarea mesajelor se realizează prin înlocuirea simbolurilor cu cuvintele binare respective. De exemplu, cuvîntul START se reprezintă în codul ASCII prin următoarea secvență de cuvinte binare:

1010011 1010100 1000001 1010010 1010100.

Evident, decodificarea se va realiza în ordine inversă. De exemplu, secvența de cuvinte binare

1010011 1010100 1001111 1010000

reprezintă în codul ASCII cuvîntul STOP.

De regulă, limbajele de programare operează nu cu cuvintele binare propriu-zise, dar cu echivalentele lor zecimale. În programele PASCAL echivalentele zecimale ale caracterelor pot fi aflate cu ajutorul funcției predefinite `ord`. De exemplu:

`ord('S')=83;` `ord('T')=84;` `ord('A')=65;` `ord('R')=82`

etc. Funcția predefinită `chr` returnează caracterul care corespunde echivalentului zecimal indicat. Astfel,

`chr(83)='S';` `chr(84)='T';` `chr(65)='A';` `chr(82)='R'.`

Orientat la textele engleze, codul ASCII nu include literele cu semne diacritice și caracterele grafice speciale întâlnite în diferite limbi europene și în lucrările științifice. De aceea pentru calculatoarele moderne s-au elaborat versiuni dedicate ale codului ASCII, denumite **coduri ASCII extinse**. Codurile extinse sînt 8-poziționale și includ $2^8 = 256$ de simboluri. Structura codurilor respective este prezentată în *tabelul 2.4*.

Partea 1 a fiecărui cod extins include simbolurile de la 0 la 127 oferite de codul ASCII. **Partea a 2-a** este specifică fiecărei țări și include simbolurile de la 128 la 255. Aceste simboluri sînt utilizate pentru reprezentarea literelor alfabetelor naționale, precum și a caracterelor științifice frecvent utilizate. Pentru exemplificare, în *tabelul 2.4* sînt prezentate codurile literelor *Ă, ă, Â, â, Î, î, Ș, ș, Ț, ț* din alfabetul limbii române propuse în anul 1992 de firma *TISH* (Chișinău). Este firesc ca utilizarea codurilor extinse să asigure prelucrarea informațiilor prezentate în diferite limbi.

Un alt exemplu de cod alfanumeric este codul binar 8-pozițional **EBCDIC** (*Extended Binary Coded Data Interchange Code*), care se utilizează pe calculatoarele mari.

Menționăm că extinderea domeniului de aplicare a codurilor 8-poziționale a favorizat utilizarea octetului și a multiplilor lui pentru măsurarea cantității de informație:

1 octet = $2^3 = 8$ biți;

1 Gigaoctet = $2^{30} \approx 10^9$ octeți;

1 Kilooctet = $2^{10} \approx 10^3$ octeți;

1 Teraoctet = $2^{40} \approx 10^{12}$ octeți;

1 Megaoctet = $2^{20} \approx 10^6$ octeți;

1 Petaoctet = $2^{50} \approx 10^{15}$ octeți.

În literatura de specialitate octetul este notat prin *B* (byte), iar multiplii respectivi prin *KB*, *MB*, *GB*, *TB* și *PB*.

Structura codurilor ASCII extinse

<i>Simbol</i>	<i>Cuvînt binar</i>	<i>Echivalent zecimal</i>	<i>Observații</i>
Spațiu	00100000	32	Partea 1: – simbolurile codului ASCII
!	00100001	33	
"	00100010	34	
#	00100011	35	
...	...	...	
}	01111101	125	
~	01111110	126	
Del	01111111	127	
A	10000000	128	Partea a 2-a: – simboluri specifice limbilor naționale; – caractere grafice; – caractere științifice
B	10000001	129	
B	10000010	130	
...	...	...	
/	11110000	240	
Ă	11110001	241	
ă	11110010	242	
Â	11110011	243	
â	11110100	244	
Î	11110101	245	
î	11110110	246	
Ș	11110111	247	
ș	11111000	248	
Ç	11111001	249	
—	11111010	250	
Ö	11111011	251	
Ƨ	11111100	252	
ţ	11111101	253	
□	11111110	254	
	11111111	255	

Întrebări și exerciții

- 1 Cîte mesaje posibile pot fi codificate cu ajutorul unui cod m -pozițional?
- 2 Determinați codurile admise de sistemul de operare cu care lucrați dvs.
- 3 Codificați în codul *Gray* următoarele șiruri de cifre zecimale: 123, 461, 952, 783, 472.
- 4 Decodificați mesajele reprezentate în codul *Aiken*:

a) 0011 1111 0100

b) 1110 0010 1101

- | | | | | | | | |
|----|------|------|------|----|------|------|------|
| c) | 1111 | 0000 | 0100 | e) | 0011 | 1100 | 1111 |
| d) | 0010 | 0001 | 1011 | f) | 1111 | 1101 | 0000 |

5 Codificați în codul ASCII expresiile:

- | | | | |
|----|--------------|----|---------|
| a) | A+B | d) | NEXT I |
| b) | FOR I=1 TO N | e) | PAUSE |
| c) | PRINT A\$ | f) | PROGRAM |

6 Decodificați mesajele reprezentate în codul ASCII:

- | | | | | | |
|----|---------|---------|----------|----------|----------|
| a) | 1000010 | 1100101 | 1100111 | 1101001 | 1101110; |
| b) | 1010011 | 1110100 | 1101111 | 1110000; | |
| c) | 1000101 | 1101110 | 1100100; | | |
| d) | 1101001 | 0111010 | 0111101 | 0110001 | 0111011. |

7 Elaborați un program care afișează pe ecran codurile următoarelor caractere, introduse de la tastatură:

- cifrele zecimale 0, 1, 2, ..., 9;
- literele latine mari A, B, C, ..., Z;
- literele latine mici a, b, c, ..., z;
- semnele operațiilor aritmetice;
- caracterele speciale ;, <, =, >, ?, [,], {, }, /, \.

2.4. Informația mesajelor continue

Sursele de informație studiate pînă în prezent se definesc printr-o variabilă S , care poate lua valori dintr-o mulțime finită de elemente distincte $\{s_1, s_2, \dots, s_n\}$, denumită mulțimea mesajelor posibile. Practica ne demonstrează că nu toate sursele de informație pot fi definite direct în acest mod. Drept exemplu vom remarca termometrele cu mercur sau alcool, vitezometrele automobilelor, microfoanele, camerele de luat vederi etc. Astfel de surse pot fi definite printr-o variabilă S (temperatura, viteza momentană, tensiunea la bornele de ieșire ale microfonului etc.) care poate lua orice valoare într-un anumit interval $[s_{\min}, s_{\max}]$.

Sursele de informație care sînt definite printr-o variabilă S ce ia valori dintr-o mulțime finită de elemente distincte se numesc **surse cu mesaje discrete**. Sursele care se definesc printr-o variabilă S ce poate lua orice valori într-un anumit interval se numesc **surse cu mesaje continue**.

Ca și în cazul surselor discrete, mesajele continue se produc în timp. Prin urmare, S este o funcție de timp, $S = S(t)$. În scopul evaluării cantității de informație în mesajele continue, vom analiza valorile funcției $S(t)$ numai în momentele de timp $t_1, t_2, \dots, t_m$ (fig. 2.4). Valorile date, notate prin $S(t_1), S(t_2), \dots, S(t_m)$, se numesc **eșantioane** (din franceză *echantillon* „cantitate mică luată dintr-un produs pentru a determina calitatea lui”).

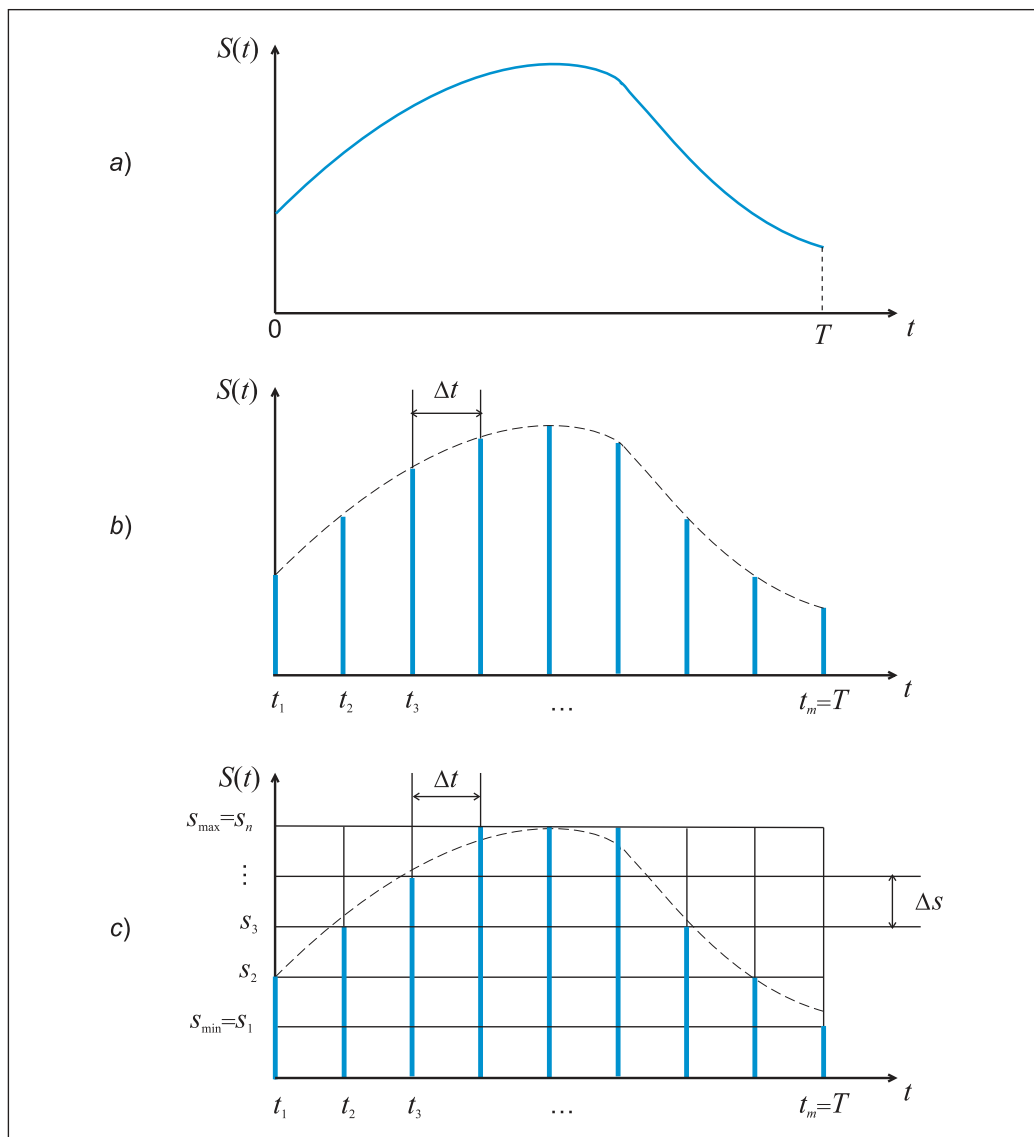


Fig. 2.4. Discretizarea mesajelor continue: a – mesaj continuu; b – mesaj discretizat în timp (eșantionat); c – mesaj eșantionat și discretizat în valoare (cuantificat)

De obicei, momentele de timp $t_1, t_2, \dots, t_m$ se definesc conform relației:

$$t_i = t_{i-1} + \Delta t.$$

Mărimea Δt se numește **perioada de eșantionare**. Valoarea concretă a perioadei de eșantionare Δt este determinată de viteza cu care $S(t)$ variază în timp.

De exemplu, în meteorologie schimbările de temperatură se produc pe parcurs de ore și $\Delta t = 1 \text{ h}$, iar în tehnica prelucrării semnalelor sonore $\Delta t = 5 \cdot 10^{-5} \text{ s}$.

Operația de transformare a mesajelor continue în eșantioane se numește discretizare în timp sau eșantionare.

Evident, pînă la recepția mesajului continuu, valorile concrete ale eșantioanelor $S(t_1), S(t_2), \dots, S(t_m)$ sînt necunoscute destinatarului. Se cunoaște numai intervalul $[s_{\min}, s_{\max}]$ în care poate lua valori arbitrare orice eșantion. În scopul evaluării cantității de informație dintr-un eșantion, vom rotunji valoarea $S(t_i)$ la una din valorile prestabilite $s_1, s_2, \dots, s_n$ (fig. 2.4).

Valorile prestabilite $s_1, s_2, \dots, s_n$ se numesc **cuante**, iar operația de transformare a valorilor curente ale mesajelor continue în cuante se numește **discretizare în valoare** sau **cuantificare**.

De obicei,

$$s_1 = s_{\min}, \quad s_2 = s_{\min} + \Delta s, \quad \dots, \quad s_i = s_{i-1} + \Delta s, \quad \dots, \quad s_n = s_{\max}.$$

Mărimea Δs reprezintă **pasul** sau **intervalul de cuantificare**. Valoarea concretă a intervalului de cuantificare depinde de natura fizică a sursei de informație, precizia de măsurare, puterea de rezoluție a destinatarului etc.

De exemplu, pentru un termometru medical $s_{\min} = 34^\circ$, $s_{\max} = 42^\circ$ și $\Delta s = 0,1^\circ$. În meteorologie $s_{\min} = -60^\circ$, $s_{\max} = +60^\circ$, $\Delta s = 1^\circ$. Pentru vitezometrul unui automobil $s_{\min} = 0$, $s_{\max} = 150$ km/h, $\Delta s = 5$ km/h.

Numărul eșantioanelor m și numărul cuantelor n se determină din următoarele relații (fig. 2.4 b și c):

$$m = \frac{T}{\Delta t} + 1; \quad n = \frac{|s_{\max} - s_{\min}|}{\Delta s} + 1,$$

unde T este durata mesajului continuu.

Întrucît cuantele $s_1, s_2, \dots, s_n$ pot fi considerate mesaje discrete, cantitatea de informație într-un eșantion

$$I = \log_2 n = \log_2 \left(\frac{|s_{\max} - s_{\min}|}{\Delta s} + 1 \right),$$

iar cantitatea de informație într-un mesaj continuu:

$$V = mI = \left(\frac{T}{\Delta t} + 1 \right) \log_2 \left(\frac{|s_{\max} - s_{\min}|}{\Delta s} + 1 \right).$$

De exemplu, pentru termometrul medical

$$I = \log_2 \left(\frac{|42 - 34|}{0,1} + 1 \right) \approx 6,34 \text{ biți},$$

iar pentru vitezometrul automobilului

$$I = \log_2 \left(\frac{|150 - 0|}{5} + 1 \right) \approx 4,95 \text{ biți}.$$

Cantitatea de informație dintr-o înregistrare audio, pentru care $n = 256$, $\Delta t = 5 \cdot 10^{-5}$ s și $T = 45$ min. este

$$V = \frac{45 \cdot 60}{5 \cdot 10^{-5}} \log_2 256 = 4,32 \cdot 10^8 \text{ biți} = 432 \text{ Mbiți}.$$

Dacă precizia de măsurare și puterea de rezoluție a destinatarului cresc, perioada de eșantionare Δt și pasul de cuantificare Δs pot fi micșorate. În consecință, va crește și cantitatea de informație dintr-un mesaj continuu.

Informația mesajelor continue poate fi reprezentată printr-un set de cuvinte binare. Pentru aceasta, cuantele $s_1, s_2, \dots, s_n$ se codifică exact la fel ca și oricare alte mesaje discrete. De exemplu, indicațiile unui termometru medical ($I \approx 6,34 \text{ biți}$) pot fi codificate cu un cod 7-pozițional. Cel mai frecvent se utilizează codurile numerice directe (tab. 2.2), cuvîntul de cod reprezentînd numărul cuantei respective. În unele aplicații se utilizează codul *Gray*, care este insensibil la perturbatii.

Dispozitivul care transformă mesajul continuu aplicat la intrare într-o succesiune de cuvinte de cod se numește **convertor analog-numeric**. Operația inversă, și anume transformarea cuvintelor de cod aplicate la intrare în valorile cuantelor respective, se efectuează cu ajutorul **convertoarelor numeric-analogice**. Utilizarea convertoarelor este necesară în cazurile în care informația de procesat este reprezentată prin mesaje continue: controlul proceselor tehnologice, dirijarea obiectelor aflate în mișcare, monitorizarea parametrilor fiziologici în medicină, filtrarea și mixajul semnalelor audio etc.

Întrebări și exerciții

- ① Care este diferența dintre sursele cu mesaje discrete și sursele cu mesaje continue?
- ② Dați cîteva exemple de surse cu mesaje continue. Concretizați intervalul în care ia valori variabila ce definește sursa.
- ③ Explicați cum se efectuează operația de eșantionare. Cum se alege perioada de eșantionare?
- ④ Explicați cum se efectuează operația de cuantificare. Cum se alege pasul de cuantificare?
- ⑤ Cum influențează valorile perioadei de eșantionare și ale pasului de cuantificare cantitatea de informație extrasă dintr-un mesaj continuu?
- ⑥ Altimetrul cu impulsuri al unui avion poate măsura înălțimi de la 100 m pînă la 20 km. Eroarea de măsurare nu depășește 1 m. Pentru a efectua o măsurare sînt necesare 10^{-3} s. Determinați cantitatea de informație furnizată de altimetru timp de 5 ore de zbor.
- ⑦ Temperatura din interiorul unui reactor chimic se înregistrează pe o bandă de hîrtie milimetrică. Pe axa absciselor se indică timpul (1 mm reprezintă o oră), iar pe axa ordonatelor – temperatura (1 mm reprezintă 10°C). Cîtă informație conține o înregistrare efectuată timp de 30 de zile, dacă temperatura poate varia de la 80 pînă la 1000°C ?
- ⑧ Pentru înregistrarea sunetului se utilizează microfoane, tensiunea de ieșire a cărora variază de la 0 pînă la $100 \mu\text{V}$. Aparatul de înregistrare nu distinge tensiunile valorilor cărora diferă cu mai puțin de $0,1 \mu\text{V}$. Pentru a asigura o reproducere fidelă, în fiecare secundă se iau 40 000 de eșantioane. Cîtă informație va furniza microfonul dat timp de 3 ore?
- ⑨ Care este destinația convertoarelor analog-numerice și numeric-analogice?

- ⑩ Elaborați un program care introduce de la tastatură valorile curente ale eșanțioanelor și afișează pe ecran codurile cuantelor corespunzătoare.
- ⑪ Elaborați un program care simulează funcționarea unui convertor numeric-analog.

2.5. Cuantizarea imaginilor

Imagine se numește reprezentarea unui obiect, executată pe o suprafață prin acțiunea directă a utilizatorului sau prin intermediul unui echipament. Cu titlul de exemplu amintim desenele, fotografiile, imaginile formate de diverse sisteme optice, optico-mecanice sau optico-electronice: microscopul, telescopul, aparatele cinematografice, televiziunea etc.

Pentru a evalua cantitatea de informație, imaginea este împărțită în **microzone**, numite de cele mai multe ori **puncte** sau **pixeli**. Descompunerea imaginii în puncte se realizează cu ajutorul unui **rastru** (de la cuvântul latin *raster* „greblă”). Rastrul reprezintă o suprafață plană, în general dreptunghiulară, pe care sînt trasate două seturi de linii paralele, perpendiculare între ele (fig. 2.5). Densitatea liniilor și, respectiv, densitatea punctelor caracterizează puterea de rezoluție a echipamentelor pentru reproducerea sau formarea imaginilor.

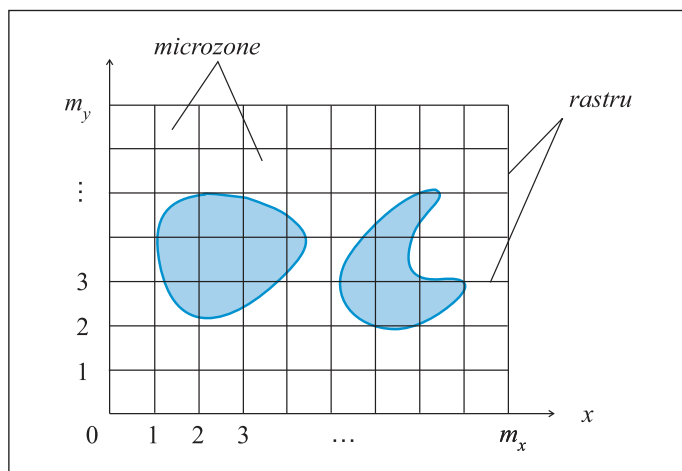


Fig. 2.5. Descompunerea imaginii în microzone

De exemplu, pentru ilustrațiile de gazetă se folosesc rastre cu rezoluția 24–30 linii/cm (576–900 de puncte pe 1 cm^2), iar pentru reproducerea tablourilor – rastre cu 54–60 linii/cm. Rastrul vizualizatorului, adică desenul pe care-l formează fasciculul de electroni pe ecranul tubului catodic, poate include 640×480 , 800×600 , 720×400 , ..., 1024×1024 de puncte.

Descompunerea imaginii în puncte (microzone) reprezintă o procedură de discretizare în spațiu.

În cazul imaginilor monocrome (alb-negru), fiecare microzonă se descrie prin **luminanța** (strălucirea) sa, care în general este o mărime continuă. Această mărime poate fi discretizată în valoare (cuantificată). Numărul cuantelor n va caracteriza puterea de rezoluție a echipamentelor pentru reproducerea sau formarea imaginilor. Prin urmare, cantitatea de informație a unei imagini monocrome:

$$I = m_x m_y \log_2 n,$$

unde m_x și m_y reprezintă numărul de microzone ale rastrului respectiv pe orizontală și verticală (fig. 2.5).

Întrucât culorile pot fi redată prin suprapunerea a trei reprezentări ale aceleiași imagini în roșu, verde și albastru, cantitatea de informație dintr-o imagine color se determină din relația:

$$I = 3 m_x m_y \log_2 n.$$

Imaginile obiectelor în mișcare se discretizează în timp, de obicei 24 (cinematograful) sau 25 (televizorul) de cadre pe secundă. Prin urmare, cantitatea de informație a unui film cu durata T se determină din relația:

$$V = T f I,$$

unde f este frecvența cadrelor, iar I cantitatea de informație dintr-un singur cadru.

De exemplu, în televiziune $m_x \approx m_y = 625$, $n = 32$ și $f = 25$ de cadre pe secundă. Un cadru color va conține:

$$I = 3 \cdot 625 \cdot 625 \cdot \log_2 32 \approx 5,6 \text{ Mbiți.}$$

Un film color cu durata de 1,5 ore va conține:

$$V = 1,5 \cdot 3\,600 \cdot 25 \cdot I \approx 791 \text{ Gbiți.}$$

Setul de cuvinte binare care reprezintă informația microzonelor se numește imagine numerică. Operația de transformare a imaginii într-un set de cuvinte binare se numește cuantizarea imaginii.

Imaginile preluate de camerele video se cuantizează cu ajutorul convertoarelor analog-numerice. Imaginile de pe hîrtie pot fi cuantizate cu ajutorul unui dispozitiv special, numit **scanner**. Acest dispozitiv conține celule fotosensibile, convertoare analog-numerice și mecanisme de avansare a hîrtiei.

Imaginile numerice se transformă în imagini propriu-zise cu ajutorul convertoarelor numeric-analogice și al echipamentelor de formare a rastrului: tubul catodic și sistemul de baleere în vizualizatoare, matricea de ace în imprimantele mecanice etc.

Întrebări și exerciții

- ❶ Numiți operațiile necesare pentru a cuantiza imaginea.
- ❷ Care este destinația rastrului? Din care considerente se alege densitatea liniilor unui rastru?
- ❸ Cum se evaluează cantitatea de informație dintr-o imagine monocromă?
- ❹ Cum pot fi redată culorile unei imagini multicolore? Cum se evaluează cantitatea de informație dintr-o imagine color?

- ⑤ Evaluați cantitatea de informație dintr-o fotografie de ziar cu dimensiunile 10×10 cm, redată cu ajutorul unui rastru ce conține 24 de puncte/cm. Fiecare punct poate avea următoarele nuanțe: alb, gri-deschis, gri-închis, negru.
- ⑥ Câtă informație conține o fotografie color cu dimensiunile 20×20 cm, reprodușă cu ajutorul unui rastru ce conține 60 de puncte/cm? Pot fi redată până la 256 de niveluri de luminanță ale punctelor respective.
- ⑦ Rastrul unei camere de luat vederi este format din 1024×1024 de puncte. Pot fi redată până la 64 de niveluri de luminanță ale punctelor respective. Câtă informație va conține un film video cu durată de 3 ore?
- ⑧ Imaginea numerică conține câte un cuvânt binar pentru fiecare punct al rastrului unui vizualizator. Câte niveluri de luminanță pot fi redată pe ecran dacă cuvintele imaginii numerice sînt 3-poziționale? 5-poziționale? 8-poziționale?

2.6. Reprezentarea și transmiterea informației

Obiectul material folosit pentru păstrarea, transmiterea sau prelucrarea informației se numește **purtător de informație**. Deosebim purtători statici și purtători dinamici de informație.

Purtătorii statici se utilizează pentru păstrarea informației sau, cu alte cuvinte, pentru transmiterea ei în timp. Informația înregistrată pe un purtător static poate fi citită în scopul prelucrării sau utilizării ulterioare. Primii purtători statici folosiți de omenire au fost pietrele, plăcile de lut ars, papirusul. Un alt purtător static de informație îl constituie hîrtia. Mesajele înregistrate pe hîrtie sub formă de manuscrise, desene sau texte tipărite pot fi păstrate un timp foarte îndelungat. În calculatoare, ca purtători statici, se utilizează:

- hîrtia pentru imprimantele mecanice, cu jet de cerneală, laser etc.;
- straturile active ale benzilor, ale discurilor magnetice;
- mediile reflectoare ale discurilor optice etc.

Transmiterea informației în spațiu se realizează cu ajutorul **purtătorilor dinamici**. În calitate de purtători dinamici tehnica actuală folosește:

- unde acustice în gaze (aer) sau lichide;
- tensiuni și curenți electrici;
- unde electromagnetice etc.

Întrucît transmiterea informației se produce în spațiu și timp, cel puțin o mărime fizică a purtătorului de informație trebuie să se schimbe.

Se numește semnal variația mărimii fizice ce asigură transmiterea mesajelor. Caracteristica semnalului folosită pentru reprezentarea (descrierea) mesajelor se numește parametru informațional al semnalului.

De exemplu, în transmisiunile radio și TV, în calitate de purtători se utilizează undele electromagnetice. Amplitudinea sau frecvența acestor unde poate varia în timp (fig. 2.6). În primul caz, parametrul informațional îl constituie amplitudinea oscilațiilor, iar în al doilea – frecvența lor.

În calculatoarele electronice în calitate de purtător de informație se folosește, de obicei, curentul electric, tensiunea și intensitatea curentului fiind para-

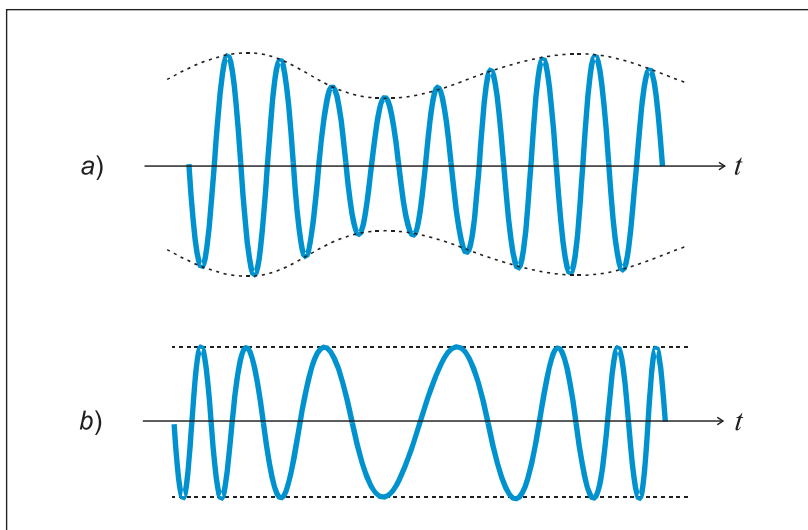


Fig. 2.6. Parametrii informaționali ai undelor electromagnetice:
a – amplitudinea; b – frecvența

metrii informaționali ai semnalului. Semnalele respective au formele prezentate în figura 2.7. În cazul nivelurilor de tensiune, unei valori a tensiunii i se asociază cifra binară 0, iar alteia cifra binară 1. Cifrele binare 0 și 1 pot fi de asemenea asociate, respectiv, cu absență sau prezență de impuls sau cu impuls negativ și impuls pozitiv.

Semnalul se numește discret, dacă parametrul informațional respectiv poate lua numai un număr finit de valori. Semnalul se numește continuu dacă parametrul informațional poate lua orice valoare într-un anumit interval.

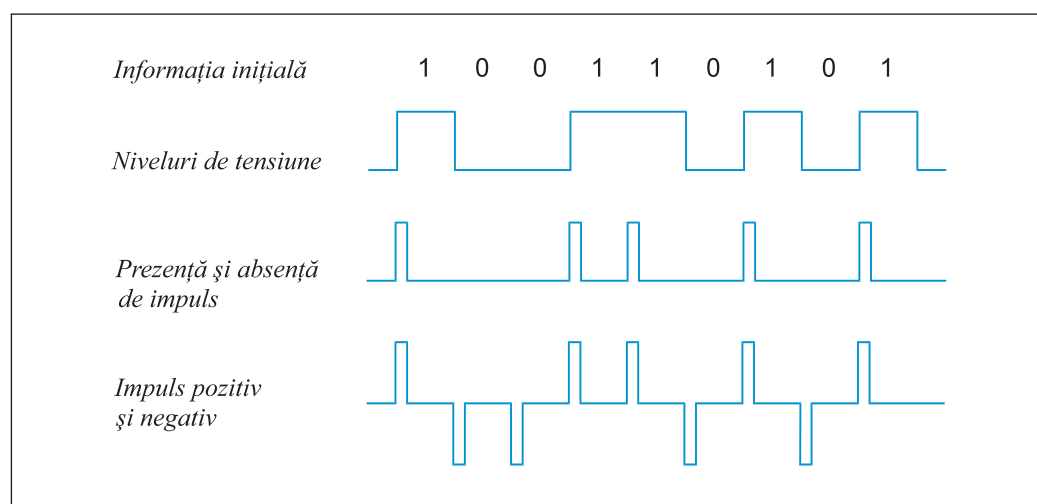


Fig. 2.7. Semnale utilizate în tehnica de calcul

De exemplu, semnalele din *figura 2.6* sînt semnale continue, iar semnalele din *figura 2.7* sînt semnale discrete. Evident, semnalele discrete și continue sînt forme de reprezentare a mesajelor respective.

Orice sistem tehnic utilizează acele semnale care-i asigură o realizare cît mai bună a funcțiilor pentru care a fost conceput. Calculatoarele actuale utilizează niveluri de tensiune, rețelele telefonice – curenți electrici, iar radioul și televiziunea – unde electromagnetice etc. Practica demonstrează că semnalele continue pot fi transmise la distanțe mult mai mari decît cele discrete. Prin urmare, pentru a asigura legătura între calculatoarele aflate la distanță, la emisie semnalele discrete vor fi transformate în semnale continue. La recepție se va realiza transformarea inversă: semnalul continuu va fi transformat într-un semnal discret.

Operația prin care parametrul informațional al semnalului continuu se modifică în funcție de valorile semnalului discret se numește modulare.

Dispozitivul tehnic care realizează operația în cauză se numește **modulator**. Pentru exemplificare, în *figura 2.8* sînt prezentate operațiile de modulare în amplitudine și frecvență.

Operația de extragere a semnalului discret dintr-un semnal continuu în funcție de procedeul de modulare adoptat se numește demodulare.

Dispozitivul tehnic care realizează operația respectivă se numește **demodulator**.

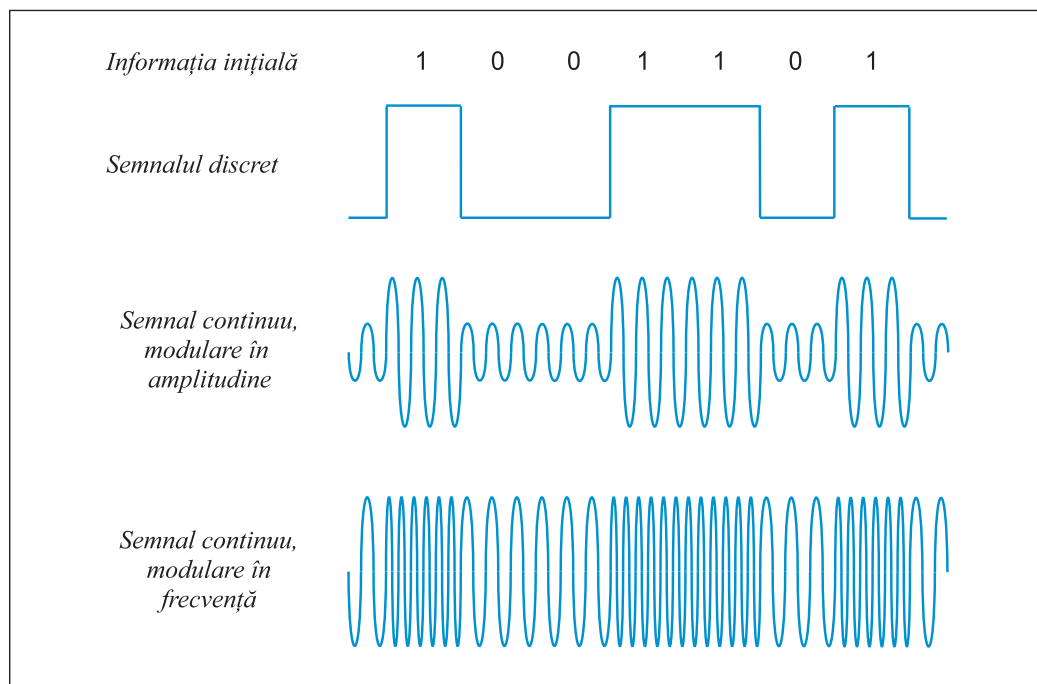


Fig. 2.8. Modularea semnalelor continue

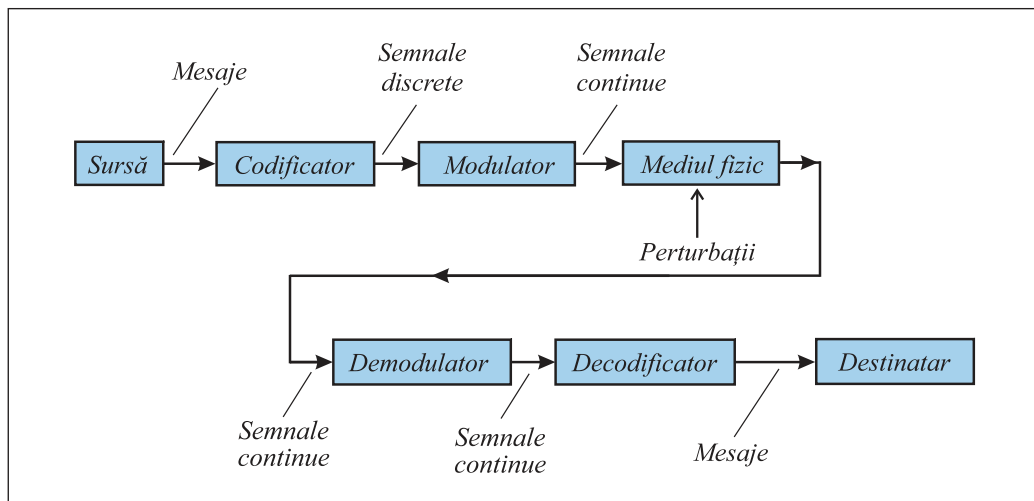


Fig. 2.9. Schema detaliată a unui sistem de transmisie a informației

Schema detaliată a sistemului de transmisie a informației este prezentată în figura 2.9.

Amintim destinația componentelor sistemului examinat:

codificatorul – transformă mesajele emise de sursă în cuvinte binare;

modulatorul – transformă semnalele discrete ce reprezintă cuvintele binare în semnale continue;

mediul fizic – reprezintă conductorii, fibrele optice, eterul etc. prin care se propagă semnalele continue;

demodulatorul – transformă semnalele continue în semnale discrete;

decodificatorul – transformă cuvintele binare în mesaje.

Evident, în cazul utilizării codurilor corectoare și a codurilor detectoare de erori, sistemul de transmisie va fi insensibil la perturbații.

Caracteristica principală a oricărui sistem de transmisie a informației este **capacitatea de transmisie**, exprimată în **biți pe secundă**. Capacitatea de transmisie depinde de caracteristicile fizice ale componentelor sistemului, metodele de modular-demodulare, caracteristicile statistice ale perturbațiilor. De exemplu, capacitatea de transmisie a unui canal telefonic este de circa 34 *Kbit/s*; capacitatea unui canal radio, cu unde centimetrice – circa 1 *Gbit/s*; capacitatea unui canal optic – 1 *Tbit/s*.

Întrebări și exerciții

- ❶ Care este deosebirea dintre purtătorii statici și purtătorii dinamici de informație?
- ❷ Determinați tipul următorilor purtători de informație:
 - a) cartele perforate;
 - b) unde ultrasonore;
 - c) benzi perforate;
 - d) unde acustice;
 - e) pelicule fotosensibile;

- f) unde gravitaționale;
g) hîrtie fotografică.
- ③ Numiți parametrii informaționali ai semnalelor emise de următoarele surse:
a) microfonul;
b) stația de radio, unde lungi, medii sau scurte;
c) instrumentul muzical;
d) stația de radio, unde ultrascurte;
e) camera de luat vederi.
- ④ Descrieți purtătorii de informație și semnalele utilizate în calculatoarele actuale.
- ⑤ Explicați operațiile de modulare și demodulare a semnalelor.
- ⑥ Care este destinația modulatorului? Dar a demodulatorului?
- ⑦ În *figura 2.8* sînt redat semnalele continue ce reprezintă cuvîntul binar 1001101. În codul ASCII (*tab. 2.3*) acestui cuvînt îi corespunde simbolul M. Redați pe un desen semnalele continue ce corespund următoarelor simboluri:
- | | |
|-------|-------|
| a) >; | e) <; |
| b) r; | f) K; |
| c) W; | g) a; |
| d) 9; | h) @. |
- ⑧ De ce depinde capacitatea de transmisie a canalului? În ce unități se măsoară ea?
- ⑨ Cum influențează perturbațiile asupra capacității de transmisie a canalului?

Test de autoevaluare nr. 2

1. Scrieți mulțimea cuvintelor 3-poziționale formate din cifrele binare {0, 1}.
2. În codul propus de filosoful englez Francis Bacon literele mari ale alfabetului latin se reprezintă astfel:

A – 00000, B – 00001, C – 00010, D – 00011, ..., Z – 11001.

Decodificați, utilizînd *codul Bacon*, cuvintele binare 00010, 00000, 00010, 00001. Codificați, utilizînd același cod, cuvîntul BAC.

3. Se consideră o sursă de informație, mulțimea de mesaje posibile ale căreia este formată din literele mari și mici ale alfabetului latin $S = \{A, B, C, \dots, Z, a, b, c, \dots, z\}$. Determinați lungimea minimă m a cuvintelor binare ce vor asigura codificarea și decodificarea univocă a mesajelor.

4. Determinați lungimea minimă m a cuvintelor binare ce vor asigura codificarea și decodificarea univocă a mesajelor numerice ale unui calendar electronic. Mesajele respective au forma $zz.ll.aa$, unde zz reprezintă ziua, ll – luna, iar aa – anul (ultimele două cifre).

5. Cîtă informație se conține într-un simbol al codului ASCII extins?

6. Considerînd că textul va fi reprezentat în codul ASCII extins, determinați cantitatea de informație într-o dictare scrisă timp de 10 min. de un elev, care este capabil să scrie circa 200 de caractere pe minut.

7. Elaborați un program PASCAL care afișează pe ecran codurile caracterelor citite de la tastatură.

8. Elaborați un program PASCAL care afișează pe ecran caracterele ce corespund codurilor citite de la tastatură.

9. Calculați câtă informație conține o fotografie color cu dimensiunile 10×10 cm, reprodușă cu ajutorul unui rastru ce conține 30 de puncte/cm. Pot fi redată până la 128 de niveluri de luminanță ale punctelor respective.

10. Selectați din lista ce urmează purtătorii statici de informație:

- | | |
|----------------------------|------------------------------|
| a) cartela perforată; | e) unda electromagnetică; |
| b) unda ultrasonoră; | f) cartela magnetică; |
| c) banda magnetică; | g) curentul electric; |
| d) pelicula fotosensibilă; | h) hîrtia pentru imprimantă. |

11*. Termometrul medical asigură măsurarea temperaturilor din intervalul $34\text{--}42^\circ$ cu o precizie de $0,1^\circ$. Pe parcursul unei zile temperatura pacientului a fost măsurată de 3 ori. Câtă informație conțin rezultatele acestor măsurări?

12*. Calculați câtă informație conține o înregistrare audio cu perioada de eșantionare $\Delta t = 3 \cdot 10^{-5}$ s, durata de $T = 30$ min. și numărul de cuante $n = 128$.

13*. Indicați corespondența între noțiunile din coloana stîngă și definițiile din coloana dreaptă:

- | | |
|-----------------|--|
| (1) cod; | (a) o știre, o comunicare verbală sau scrisă; |
| (2) semnal; | (b) o mulțime de numere binare; |
| (3) informație; | (c) o mulțime ordonată de semne; |
| (4) alfabet; | (d) un anumit număr de biți; |
| | (e) regula de transformare a mesajelor în cuvinte binare; |
| | (f) variația mărimii fizice ce asigură transmiterea mesajelor. |

14*. Indicați corespondența între denumirile componentelor (coloana din stînga) ale unui sistem de transmisie a informației și destinația acestora (coloana din dreapta):

- | | |
|--------------------|---|
| (1) codicatorul; | (a) transformă semnale continue în semnale discrete; |
| (2) modulatorul; | (b) transformă secvențele sonore în cuvinte binare; |
| (3) demodulatorul; | (c) transformă mesajele emise de sursă în cuvinte binare; |
| (4) decodicatorul; | (d) transformă imaginile grafice în cuvinte binare; |
| | (e) transformă cuvintele binare în mesaje; |
| | (f) transformă semnalele discrete în semnale continue. |

* Numai pentru profilul real.

BAZELE ARITMETICE ALE TEHNICII DE CALCUL

3.1. Sisteme de numerație

În calculatoarele digitale informația de orice categorie este reprezentată, stocată și prelucrată în formă numerică. Numerele se reprezintă prin simboluri elementare denumite **cifre**.

Totalitatea regulilor de reprezentare a numerelor, împreună cu mulțimea cifrelor poartă denumirea de sistem de numerație. Numărul cifrelor definește baza sistemului de numerație.

Prezentăm câteva exemple de sisteme de numerație:

- **sistemul zecimal** este un sistem de numerație în baza 10, numărul de cifre utilizate fiind 10, respectiv, 0, 1, 2, ..., 9;
- **sistemul binar** este un sistem de numerație în baza 2, numărul de cifre utilizate este 2, adică 0 și 1. Cifrele respective se numesc **cifre binare** sau **biți**;
- **sistemul ternar** este un sistem de numerație în baza 3, numărul de cifre utilizate fiind 3, respectiv, 0, 1 și 2;
- **sistemul octal** este un sistem de numerație în baza 8, conținând 8 cifre: 0, 1, 2, ..., 7;
- **sistemul hexazecimal** este un sistem de numerație în baza 16 și conține 16 cifre: 0, 1, 2, ..., 9, *A* (zece), *B* (unsprezece), *C* (doisprezece), *D* (treisprezece), *E* (paisprezece), *F* (cincisprezece).

În *tabelul 3.1* sînt date reprezentările unora și aceluiași numere în diferite baze.

Tabelul 3.1

Reprezentarea unor numere în diferite baze

<i>Zecimal</i>	<i>Binar</i>	<i>Octal</i>	<i>Hexazecimal</i>	<i>Zecimal</i>	<i>Binar</i>	<i>Octal</i>	<i>Hexazecimal</i>
0	0	0	0	7	111	7	7
1	1	1	1	8	1000	10	8
2	10	2	2	9	1001	11	9
3	11	3	3	10	1010	12	A
4	100	4	4	11	1011	13	B
5	101	5	5	12	1100	14	C
6	110	6	6	13	1101	15	D

<i>Zecimal</i>	<i>Binar</i>	<i>Octal</i>	<i>Hexa-zecimal</i>
14	1110	16	E
15	1111	17	F
16	10000	20	10
17	10001	21	11
18	10010	22	12
19	10011	23	13
20	10100	24	14
...	...	...	...
30	11110	36	1E
...	...	...	...
40	101000	50	28
...	...	...	...

<i>Zecimal</i>	<i>Binar</i>	<i>Octal</i>	<i>Hexa-zecimal</i>
50	110010	62	32
...	...	...	...
60	111100	74	3C
...	...	...	...
70	1000110	106	46
...	...	...	...
80	1010000	120	50
...	...	...	...
90	1011010	132	5A
...	...	...	...
100	1100100	144	64
...	...	...	...

Regula de reprezentare a numerelor în sistemul zecimal rezultă din următorul exemplu:

$$(3856,43)_{10} = 3 \cdot 10^3 + 8 \cdot 10^2 + 5 \cdot 10^1 + 6 \cdot 10^0 + 4 \cdot 10^{-1} + 3 \cdot 10^{-2}.$$

Se observă că în această reprezentare semnificația (valoarea) fiecărei cifre depinde de poziția pe care o ocupă în număr. De exemplu, cifra 3 se întâlnește de 2 ori: prima dată are semnificația 3000, iar a doua oară – semnificația 0,03.

Sistemele în care semnificația cifrelor depinde de poziția ocupată în cadrul numerelor se numesc sisteme de numerație poziționale.

Presupunem că numărul N are partea întreagă formată din $n+1$ cifre, iar partea fracționară – din m cifre:

$$N = c_n c_{n-1} \dots c_1 c_0, c_{-1} c_{-2} \dots c_{-m}.$$

Valoarea acestui număr se evaluează în funcție de baza sistemului:

$$(N)_b = c_n b^n + c_{n-1} b^{n-1} + \dots + c_1 b^1 + c_0 b^0 + c_{-1} b^{-1} + c_{-2} b^{-2} + \dots + c_{-m} b^{-m}.$$

Efectuînd calculele respective, se va realiza **conversiunea** numărului $(N)_b$ din baza b în sistemul zecimal.

De exemplu,

$$(101,1)_{10} = 1 \cdot 10^2 + 0 \cdot 10^1 + 1 \cdot 10^0 + 1 \cdot 10^{-1} = 101,1;$$

$$(101,1)_2 = 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 + 1 \cdot 2^{-1} = 5,5;$$

$$(101,1)_3 = 1 \cdot 3^2 + 0 \cdot 3^1 + 1 \cdot 3^0 + 1 \cdot 3^{-1} = 10,333\dots;$$

$$(101,1)_8 = 1 \cdot 8^2 + 0 \cdot 8^1 + 1 \cdot 8^0 + 1 \cdot 8^{-1} = 65,125;$$

$$(101,1)_{16} = 1 \cdot 16^2 + 0 \cdot 16^1 + 1 \cdot 16^0 + 1 \cdot 16^{-1} = 257,0625.$$

Formal, sistemul zecimal nu prezintă niciun avantaj deosebit față de alte sisteme de numerație. Se presupune că acest sistem a fost adoptat încă din cele mai vechi timpuri datorită faptului că procesul de numărare a folosit ca instrumente inițiale degetele mâinilor.

Un calculator poate fi prevăzut să funcționeze în orice sistem de numerație. Pe parcursul dezvoltării tehnicii de calcul, s-a stabilit că cel mai avantajos este sistemul binar. Acest sistem a fost preferat din următoarele motive:

- simplitatea regulilor pentru operațiile aritmetice și logice;
- materializarea fizică a cifrelor în procesul prelucrării sau stocării numerelor se face mai ușor pentru două simboluri decât pentru zece: perforat–neperforat, contact închis–contact deschis, prezență sau absență de curent etc.;
- circuitele care trebuie să diferențieze numai între două stări sînt mai sigure în funcționare decât cele care trebuie să diferențieze între zece stări.

Menționăm că în procesul dezvoltării civilizației umane au fost create și **sisteme de numerație nepoziționale**. Drept exemplu poate servi **sistemul roman**, care utilizează cifrele *I, V, X, L, C, D, M*. Întrucît regulile de reprezentare a numerelor și de efectuare a operațiilor aritmetice sînt foarte complicate, sistemele nepoziționale au o utilizare foarte restrînsă.

Întrebări și exerciții

- ❶ Cum se definește un sistem de numerație?
- ❷ Care este deosebirea dintre sistemele de numerație poziționale și nepoziționale?
- ❸ Dați exemple de sisteme de numerație poziționale. Cum se definește baza sistemului de numerație?
- ❹ Evaluați numărul $(101,1)_b$ scris în următoarele sisteme de numerație:
 $b = 4, 5, 6, 7, 9, 11, 12, 13, 14$ și 15 .
- ❺ Evaluați numerele ce urmează:

<i>a)</i> $(328)_9$;	<i>i)</i> $(1010,01)_3$;	<i>q)</i> $(341,02)_8$;
<i>b)</i> $(516)_7$;	<i>j)</i> $(201,12)_4$;	<i>r)</i> $(ABCD)_{16}$;
<i>c)</i> $(1010,01)_2$;	<i>k)</i> $(341,02)_6$;	<i>s)</i> $(328)_{16}$;
<i>d)</i> $(201,12)_3$;	<i>l)</i> $(1111)_{16}$;	<i>t)</i> $(516)_{16}$;
<i>e)</i> $(341,02)_5$;	<i>m)</i> $(328)_{11}$;	<i>u)</i> $(1010,01)_{16}$;
<i>f)</i> $(FFFF)_{16}$;	<i>n)</i> $(516)_9$;	<i>v)</i> $(201,12)_{16}$;
<i>g)</i> $(328)_{10}$;	<i>o)</i> $(1010,01)_8$;	<i>w)</i> $(341,02)_{12}$;
<i>h)</i> $(516)_8$;	<i>p)</i> $(201,12)_8$;	<i>x)</i> $(F001)_{16}$.
- ❻ Care factori au contribuit la utilizarea în tehnica de calcul a sistemului binar?
- ❼ Elaborați un program PASCAL care evaluează numerele scrise în baza b , $b \leq 10$.
- ❽ Elaborați un program care evaluează numerele scrise în baza b , $10 \leq b \leq 36$.

3.2. Conversiunea numerelor dintr-un sistem în altul

Conversiunea numărului $(N)_b$ în echivalentul său zecimal se efectuează conform formulei din paragraful precedent:

$$(N)_b = c_n b^n + c_{n-1} b^{n-1} + \dots + c_1 b^1 + c_0 b^0 + c_{-1} b^{-1} + c_{-2} b^{-2} + \dots + c_{-m} b^{-m}.$$

Conversiunea numărului zecimal $(N)_{10}$ în echivalentul său în baza b se efectuează conform următoarelor reguli:

- se împarte la baza respectivă partea întreagă și cîturile obținute după fiecare împărțire, pînă se obține cîtul zero; rezultatul conversiunii părții întregi este constituit din resturile obținute, considerate în ordinea inversă de apariție;

- se înmulțește cu baza partea fracționară, apoi toate părțile fracționare obținute din produsul anterior, pînă cînd partea fracționară a unui produs este zero sau pînă la obținerea unui număr de cifre fracționare dorit; rezultatul conversiunii părții fracționare este constituit din părțile întregi ale produselor, considerate în ordinea apariției.

Să analizăm cîteva exemple.

1) Să se transforme numărul zecimal 53,40625 în echivalentul său binar.

$$53 : 2 = 26 + \frac{1}{2};$$

$$26 : 2 = 13 + \frac{0}{2};$$

$$13 : 2 = 6 + \frac{1}{2};$$

$$6 : 2 = 3 + \frac{0}{2};$$

$$3 : 2 = 1 + \frac{1}{2};$$

$$1 : 2 = 0 + \frac{1}{2}.$$

Prin urmare, partea întreagă a numărului binar va fi 110101.

$$0,40625 \times 2 = 0,8125;$$

$$0,8125 \times 2 = 1,625;$$

$$0,625 \times 2 = 1,25;$$

$$0,25 \times 2 = 0,5;$$

$$0,5 \times 2 = 1,0.$$

Partea fracționară a numărului binar va fi 01101. Prin urmare,

$$(53,40625)_{10} = (110101,01101)_2.$$

2) Să se transforme numărul 23,7 din sistemul zecimal în sistemul binar.

$$23 : 2 = 11 + \frac{1}{2};$$

$$11 : 2 = 5 + \frac{1}{2};$$

$$5 : 2 = 2 + \frac{1}{2};$$

$$2 : 2 = 1 + \frac{0}{2};$$

$$1 : 2 = 0 + \frac{1}{2};$$

$$0,7 \times 2 = 1,4;$$

$$0,4 \times 2 = 0,8;$$

$$\begin{aligned}0,8 \times 2 &= 1,6; \\ 0,6 \times 2 &= 1,2; \\ 0,2 \times 2 &= 0,4; \\ 0,4 \times 2 &= 0,8; \\ &\dots\end{aligned}$$

Se observă că operația poate fi continuată la infinit, adică nu există o conversiune exactă a numărului analizat. Prin urmare,

$$(23,7)_{10} = (10111,101100\dots)_2.$$

3) Să se efectueze conversiunea numărului 1996,0625 din sistemul zecimal în sistemul octal.

$$\begin{aligned}1996 : 8 &= 249 + ^4/8; \\ 249 : 8 &= 31 + ^1/8; \\ 31 : 8 &= 3 + ^7/8; \\ 3 : 8 &= 0 + ^3/8;\end{aligned}$$

$$\begin{aligned}0,0625 \times 8 &= 0,5; \\ 0,5 \times 8 &= 4.\end{aligned}$$

Prin urmare,

$$(1996,0625)_{10} = (3714,04)_8.$$

4) Să se transforme numărul 2914,25 din sistemul zecimal în sistemul hexazecimal.

$$\begin{aligned}2914 : 16 &= 182 + ^2/16; \\ 182 : 16 &= 11 + ^6/16; \\ 11 : 16 &= 0 + ^{11}/16;\end{aligned}$$

$$0,25 \times 16 = 4.$$

Prin urmare,

$$(2914,25)_{10} = (B62,4)_{16}.$$

Întrebări și exerciții

- ❶ Cum se efectuează conversiunea unui număr dintr-un sistem cu baza b în sistemul zecimal?
- ❷ Transformați în sistemul zecimal numerele ce urmează:

a) $(100001,01111)_2;$	e) $(AAA,BBB)_{16};$	i) $(124,521)_7;$
b) $(328,678)_9;$	f) $(EE,00F)_{16};$	j) $(4231,124)_5;$
c) $(100,100)_{16};$	g) $(1221,1112)_3;$	k) $(50,505050)_6;$
d) $(10,01)_{16};$	h) $(1321,1312)_4;$	l) $(7777,001)_8.$

③ Cum se efectuează conversiunea unui număr zecimal în echivalentul său în baza b ?

④ Transformați în sistemul binar numerele zecimale ce urmează:

- | | | |
|------------|------------|------------|
| a) 13,889; | e) 93,447; | i) 58,749; |
| b) 38,668; | f) 70,212; | j) 4,345; |
| c) 53,536; | g) 8,347; | k) 3,156; |
| d) 29,261; | h) 39,764; | l) 91,428. |

Verificați rezultatele efectuând conversiunea binar-zecimală.

⑤ Transformați în sistemul octal numerele zecimale ce urmează:

- | | | |
|-------------|-------------|-------------|
| a) 358,932; | e) 886,526; | i) 795,128; |
| b) 479,093; | f) 971,258; | j) 680,895; |
| c) 591,241; | g) 515,914; | k) 256,453; |
| d) 649,113; | h) 347,607; | l) 838,261. |

Verificați rezultatele efectuând conversiunea octal-zecimală.

⑥ Transformați în sistemul hexazecimal numerele zecimale ce urmează:

- | | | |
|--------------|--------------|--------------|
| a) 1424,699; | e) 5818,961; | i) 4985,995; |
| b) 3517,315; | f) 9336,491; | j) 9721,678; |
| c) 9607,201; | g) 3442,722; | k) 5292,837; |
| d) 8974,664; | h) 4521,449; | l) 2734,592. |

Verificați rezultatele, efectuând conversiunea hexazecimal-zecimală.

⑦ Elaborați un program care transformă numerele zecimale în echivalentele respective din sistemul de numerație în baza b , $b < 10$.

⑧ Elaborați un program pentru transformarea numerelor zecimale în numere din sistemul în baza b , $10 < b \leq 36$.

3.3. Conversiunea din binar în octal, hexazecimal și invers

Întrucât $8 = 2^3$, **conversiunea binar-octală** și **octal-binară** se poate face direct. Orice cifră octală se reprezintă prin 3 cifre binare:

0 = 000;	4 = 100;
1 = 001;	5 = 101;
2 = 010;	6 = 110;
3 = 011;	7 = 111.

Dacă este dat un număr octal, pentru conversiunea lui în binar se va scrie fiecare cifră octală prin 3 cifre binare.

Exemple:

$$(247,315)_8 = (010\ 100\ 111, 011\ 001\ 101)_2;$$

$$(512,07)_8 = (101\ 001\ 010, 000\ 111)_2;$$

$$(3,146)_8 = (011, 001\ 100\ 110)_2.$$

Dacă este considerat un număr binar, pentru conversiunea lui în octal se vor grupa câte 3 cifre binare pornind de la poziția virgulei spre stînga pentru partea întreagă, respectiv dreapta pentru partea fracționară, aflînd corespondentul în octal. Pentru completarea unui grup de trei cifre binare, zerourile din fața numărului, respectiv după ultima cifră a părții fracționare, nu modifică semnificația numărului.

Exemple:

$$(11,011101)_2 = (011,011\ 101)_2 = (3,35)_8;$$

$$(10,11011)_2 = (010,110\ 110)_2 = (2,66)_8;$$

$$(1001,01011)_2 = (001\ 001,010\ 110)_2 = (11,26)_8.$$

În mod similar se procedează și în cazul sistemului hexazecimal, baza căruia este $16 = 2^4$. Orice cifră hexazecimală se reprezintă prin 4 cifre binare:

$$0 = 0000;$$

$$8 = 1000;$$

$$1 = 0001;$$

$$9 = 1001;$$

$$2 = 0010;$$

$$A = 1010;$$

$$3 = 0011;$$

$$B = 1011;$$

$$4 = 0100;$$

$$C = 1100;$$

$$5 = 0101;$$

$$D = 1101;$$

$$6 = 0110;$$

$$E = 1110;$$

$$7 = 0111;$$

$$F = 1111.$$

Exemple:

$$(6AF3,B2)_{16} = (0110\ 1010\ 1111\ 0011, 1011\ 0010)_2;$$

$$(6F1,3CA)_{16} = (0110\ 1111\ 0001, 0011\ 1100\ 1010)_2;$$

$$(11,01101)_2 = (0011, 0110\ 1000)_2 = (3,68)_{16};$$

$$(10001,01011)_2 = (0001\ 0001, 0101\ 1000)_2 = (11,58)_{16}.$$

Întrebări și exerciții

- ❶ Cum se efectuează conversiunea octal-binară și binar-octală?
- ❷ Transformați în sistemul binar numerele octale ce urmează:
 - a) 15,006;
 - b) 13,06;
 - c) 43,15;
 - d) 10,01;
 - e) 21,626;
 - f) 6,3415;
 - g) 771,25;
 - h) 12,121;
 - i) 42,322;
 - j) 44,523;
 - k) 32,271;
 - l) 73,536.
- ❸ Transformați în sistemul octal numerele binare ce urmează:
 - a) 1,1;
 - b) 101,10101;
 - c) 1111,000101;
 - d) 10101110,1001;
 - e) 1101,1;
 - f) 1,000001;
 - g) 11110001,101;
 - h) 0,00001;
 - i) 10110,001011;
 - j) 11111,0010001;
 - k) 11001,00101;
 - l) 1011,0001011.
- ❹ Elaborați un program pentru conversiunea binar-octală și octal-binară.
- ❺ Cum se efectuează conversiunea hexazecimal-binară și binar-hexazecimală?
- ❻ Transformați în sistemul binar numerele hexazecimale ce urmează:
 - a) FFF,AAA;
 - b) F,1A;
 - c) 1,009;
 - d) 0,00F;
 - e) 3,1AB;
 - f) AABBB,0000F;
 - g) 81,91A;
 - h) 10,01;
 - i) C,DC1;
 - j) 942,14A;
 - k) CAA,B;
 - l) DAD,ABA.
- ❼ Transformați în sistemul hexazecimal numerele binare ce urmează:
 - a) 1001011101,01101;
 - b) 111,01;
 - c) 1,1;
 - d) 10101110111,00101;
 - e) 1011101,011;
 - f) 11,001011101;
 - g) 11110,01111;
 - h) 111011,0010000;
 - i) 1011,0101;
 - j) 11001,0110;
 - k) 0,00001;
 - l) 101,01011.
- ❽ Elaborați un program pentru conversiunea binar-hexazecimală și hexazecimal-binară.
- ❾ Transformați în sistemul hexazecimal numerele octale ce urmează:
 - a) 13,146;
 - b) 613,12;
 - c) 541,723;
 - d) 104,527;
 - e) 451,35;
 - f) 12,357;
 - g) 53,627;
 - h) 572,004;
 - i) 644,031;
 - j) 5,4312;
 - k) 675,542;
 - l) 372,271.

⑩ Transformați în sistemul octal numerele hexazecimale ce urmează:

a) AA;

e) 15F,6A1;

i) BBB;

b) A2B,1F;

f) 3,281;

j) AF,31B;

c) F,3A;

g) 9,AF;

k) 2C,ACB;

d) FFFF;

h) 3,418F;

l) A1B,39E.

⑪ Elaborați un program pentru conversiunea octal-hexazecimală și hexazecimal-octală.

3.4. Operații aritmetice în binar

Operațiile aritmetice cu numerele binare sînt foarte simple. Regulile de opera-re în sistemul binar sînt prezentate în *tabelele 3.2, 3.3 și 3.4*.

Exemple:

Tabelul 3.2

Adunarea binară

$0 + 0 = 0$
$0 + 1 = 1$
$1 + 0 = 1$
$1 + 1 = 10$

Tabelul 3.3

Scăderea binară

$0 - 0 = 0$
$1 - 0 = 1$
$1 - 1 = 0$
$10 - 1 = 1$

Tabelul 3.4

Înmulțirea binară

$0 \times 0 = 0$
$0 \times 1 = 0$
$1 \times 0 = 0$
$1 \times 1 = 1$

Exemple:

1) Se propune adunarea în binar a numerelor zecimale 29 și 43:

$$(29)_{10} = (11101)_2;$$

$$(43)_{10} = (101011)_2;$$

$$\begin{array}{r} 11101 \\ + 101011 \\ \hline 1001000 \end{array}$$

Verificare: $(1001000)_2 = (72)_{10}$, rezultatul este corect, întrucît $(29)_{10} + (43)_{10} = (72)_{10}$.

2) Se propune scăderea în binar a numărului zecimal 37 din numărul zecimal 46:

$$(37)_{10} = (100101)_2;$$

$$(46)_{10} = (101110)_2;$$

$$\begin{array}{r} 101110 \\ - 100101 \\ \hline 1001 \end{array}$$

Verificare: $(1001)_2 = (9)_{10}$, rezultatul este corect, întrucît $(46)_{10} - (37)_{10} = (9)_{10}$.

3) Se propune înmulțirea în binar a numerelor zecimale 3,25 și 7,125:

$$(3,25)_{10} = (11,01)_2;$$

$$(7,125)_{10} = (111,001)_2;$$

$$\begin{array}{r} 11,01 \times \\ 111,001 \\ \hline 1101 \\ 0000 \\ 0000 \\ 1101 \\ 1101 \\ 1101 \\ \hline 10111,00101 \end{array}$$

Verificare: $(10111,00101)_2 = (23,15625)_{10}$, rezultatul este corect, întrucît $(3,25)_{10} \times (7,125)_{10} = (23,15625)_{10}$.

4) Se propune împărțirea în binar a numărului zecimal 211 la numărul zecimal 3:

$$(211)_{10} = (11010011)_2;$$

$$(3)_{10} = (11)_2;$$

$$\begin{array}{r} 11010011 \quad | \quad 11 \\ 11 \quad \hline 00100 \\ 11 \quad \hline 11 \\ 11 \quad \hline 01 \end{array}$$

Deci $11010011 : 11 = 1000110$, rest 1.

Verificare: $(1000110)_2 = (70)_{10}$, rezultatul este corect, întrucît $(211)_{10} : (3)_{10} = (70)_{10} + (1)_{10}$.

Menționăm că atât la înmulțire, cât și la împărțire, fixarea virgulei care desparte partea întreagă de cea fracționară se face la fel ca și în sistemul de numerație zecimal.

Întrebări și exerciții

❶ Cum se execută operațiile aritmetice în sistemul binar?

❷ Calculați în sistemul binar:

a) $34 + 251;$

d) $14 \times 8;$

g) $2015 + 1995;$

b) $68 - 7;$

e) $63 : 3;$

h) $28,5 + 0,75;$

c) $1512 + 620;$

f) $1996 - 51;$

i) $63,125 - 4,125;$

j) $3,0625 \times 2,125;$	n) $32 : 2;$	r) $401 \times 4;$
k) $0,5 \times 0,5;$	o) $32 : 16;$	s) $32 : 4;$
l) $1 : 0,5;$	p) $401 \times 8;$	t) $401 \times 2;$
m) $40 : 0,125;$	q) $32 : 8;$	u) $933 : 3.$

Numerele analizate sînt scrise în sistemul zecimal.

- ③ Elaborați un program pentru adunarea și scăderea numerelor binare.
- ④ Elaborați un program pentru înmulțirea și împărțirea numerelor binare.

3.5. Reprezentarea numerelor naturale în calculator

Calculatoarele actuale utilizează sistemul de numerație binar. Reprezentarea numerelor naturale $N = \{0, 1, 2, \dots\}$ se realizează pe un număr fix de poziții binare, de regulă, 8, 16, 32 sau 64 (fig. 3.1).

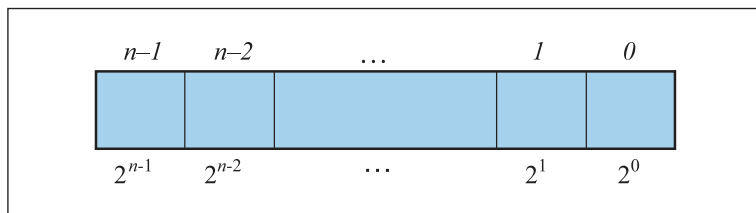


Fig. 3.1. Reprezentarea numerelor naturale pe n poziții binare

În pozițiile 0, 1, ..., $n-1$ sînt înscrise cifrele binare ale numărului natural reprezentat în sistemul de numerație binar. Alinierea numerelor binare se realizează la dreapta, eventualele zerouri nesemnificative sînt plasate în fața numărului binar.

Drept exemplu, în figura 3.2 este redată reprezentarea numărului natural

$$1039 = (10000001111)_2$$

pe un număr de 16 poziții binare.

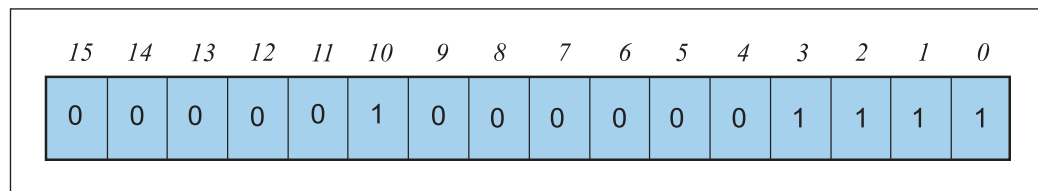


Fig. 3.2. Reprezentarea numărului natural 1039 pe 16 poziții binare

Numărul maxim ce poate fi reprezentat pe n poziții binare (fig. 3.3) este

$$1 \cdot 2^0 + 1 \cdot 2^1 + 1 \cdot 2^2 + \dots + 1 \cdot 2^{n-1} = 2^n - 1.$$

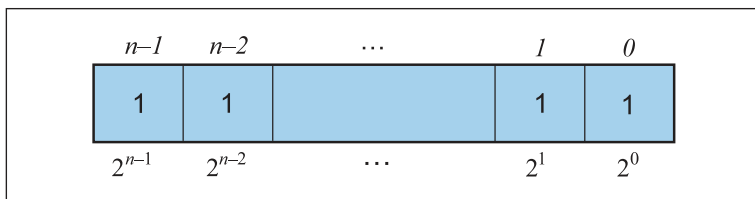


Fig. 3.3. Reprezentarea numărului natural maxim pe n poziții binare

Prin urmare, pe n poziții binare pot fi reprezentate numere naturale din intervalul $[0; 2^n - 1]$.

Întrebări și exerciții

- ❶ Cum se reprezintă numerele naturale în calculator?
- ❷ Reprezentați numerele naturale 3, 112, 191, 204, 255 pe 8 poziții binare.
- ❸ Reprezentați numerele naturale 3, 255, 1024, 2048, 4096, 65 535 pe 16 poziții binare.
- ❹ Calculați numerele naturale maxime ce pot fi reprezentate pe 4, 8, 12, 16, 24, 32 și 64 de poziții binare.

3.6. Reprezentarea numerelor întregi

În calculator nu există posibilitatea introducerii directe a semnelor $+$ și $-$, atașate numerelor pozitive și negative. Din acest motiv, reprezentarea semnului numărului x se face cu ajutorul unei cifre binare, denumită **cifra-semn**, așezată în poziția $n-1$ (fig. 3.4):

$$S = \begin{cases} 0, & x > 0; \\ 1, & x < 0. \end{cases}$$



Fig. 3.4. Reprezentarea numerelor întregi pe n poziții binare

Numerele cu semn se reprezintă cel mai des nu în sistemul binar direct, ci într-un sistem binar codificat care oferă anumite avantaje în executarea operațiilor aritmetice cu numere algebrice. Din acest punct de vedere, se cunosc trei moduri de reprezentare, denumite **coduri binare pentru numere algebrice**.

Codul direct (codul mărime și semn). Scrierea unui număr în acest cod este foarte simplă: în cifra-semn se scrie 0, dacă numărul este pozitiv, și 1, dacă el este negativ; în partea de valoare se înscrie numărul în sistemul binar obișnuit.

Drept exemplu, în *figura 3.5* sînt redate reprezentările numerelor +52 și -52 pe 8 poziții binare.

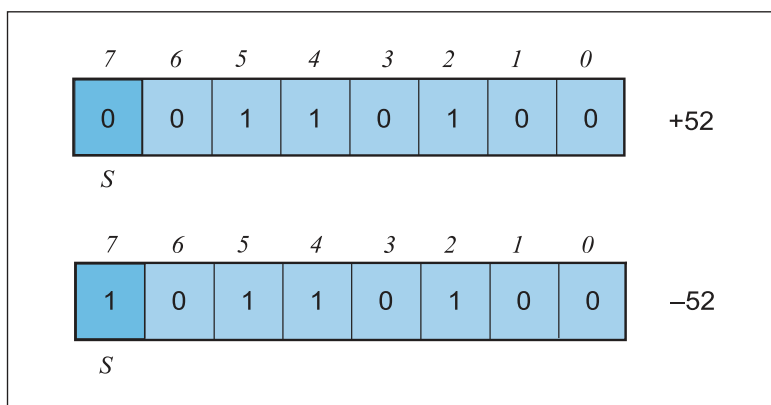


Fig. 3.5. Reprezentarea numerelor +52 și -52 în cod direct

În codul direct, pe n poziții binare se pot reprezenta numere întregi pozitive și negative N , astfel încît:

$$-(2^{n-1} - 1) \leq N \leq (2^{n-1} - 1).$$

Menționăm că în codul direct există două reprezentări binare pentru numărul 0: 00...0 și 10...0. Codul direct este rar utilizat în calculatoare, deoarece necesită algoritmi complicați de executare a operațiilor aritmetice și verificare a rezultatelor.

Codul invers. Pentru numerele pozitive scrierea în cod invers este identică cu cea din codul direct. Dacă numărul este negativ, el se scrie mai întîi ca și cum ar fi pozitiv, apoi se inversează fiecare cifră binară, adică 1 devine 0 și 0 devine 1.

În *figura 3.6* sînt redate reprezentările numerelor +52 și -52 în cod invers pe 8 poziții binare.

Se poate ușor constata că intervalul numerelor întregi N care se pot reprezenta în cod invers este același ca și pentru reprezentarea în cod direct, iar numărul 0 are două reprezentări binare: 00...0 și 11...1.

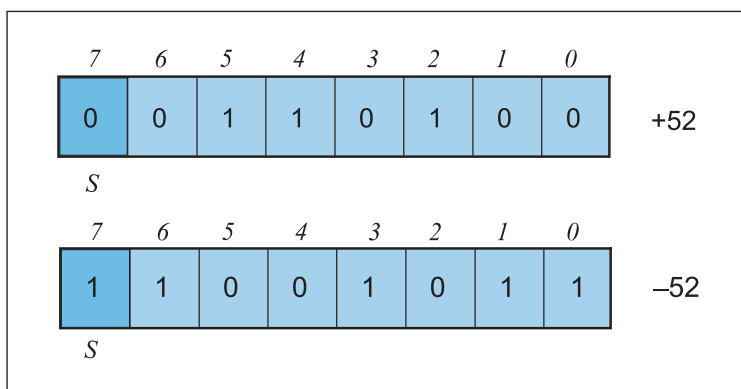


Fig. 3.6. Reprezentarea numerelor +52 și -52 în cod invers

Codul complementar. În acest cod numerele pozitive au aceeași reprezentare ca și în codul direct și codul invers. Dacă numărul este negativ, el se scrie mai întâi în codul invers, apoi se adună 1 la cifra cea mai puțin semnificativă (poziția binară 0).

Drept exemplu, în *figura 3.7* sînt redată reprezentările numerelor +52 și -52 în cod complementar.

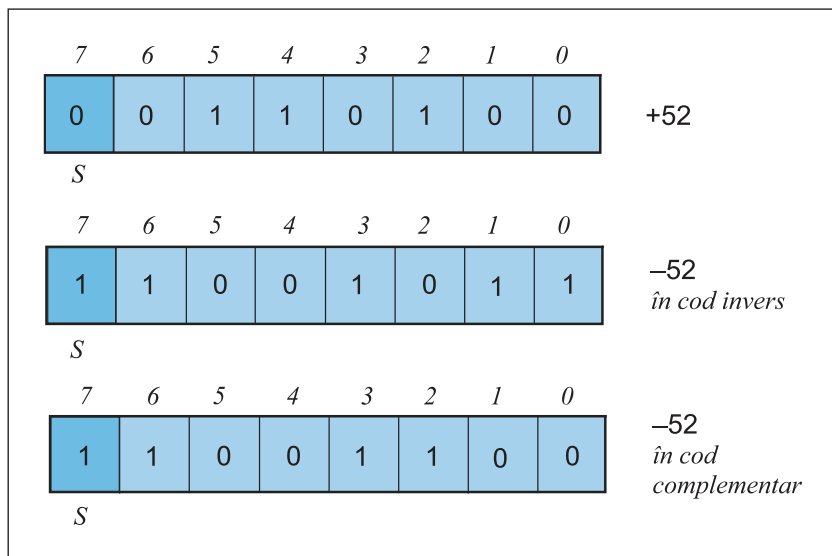


Fig. 3.7. Reprezentarea numerelor +52 și -52 în cod complementar

Codul complementar este utilizat în marea majoritate a calculatoarelor datorită facilităților pe care le oferă la efectuarea operațiilor aritmetice și ușurinței cu care se face verificarea rezultatelor. În acest cod pe n poziții binare pot fi reprezentate numere întregi din intervalul $[-2^{n-1}, 2^{n-1} - 1]$.

Întrebări și exerciții

- ❶ Cum pot fi reprezentate în calculator numerele întregi? Explicați cum se scriu numerele negative în cod direct, cod invers și cod complementar.
- ❷ Reprezentați în cod direct pe 8 poziții binare:

a) +12;	d) -12;	g) +21;
b) -21;	e) -64;	h) -68;
c) +68;	f) +105;	i) -112.
- ❸ Reprezentați în cod invers pe 8 poziții binare:

a) +10;	d) -10;	g) +65;
b) -65;	e) +101;	h) -101;
c) +112;	f) -112;	i) -105.

④ Reprezentați în cod complementar pe 8 poziții binare:

a) +40;

e) -40;

i) +16;

b) +27;

f) -27;

j) +101;

c) +109;

g) -109;

k) +111;

d) -16;

h) -101;

l) -111.

⑤ Elaborați un program care afișează pe ecran reprezentările în cod direct, cod invers și cod complementar ale numerelor întregi introduse de la tastatură pentru $n = 8, 16$ și 32 .

⑥ Cum se reprezintă numărul 0 în cod direct, cod invers și cod complementar? Câte reprezentări are numărul 0 în codurile examinate?

⑦ Determinați numărul maxim ce poate fi reprezentat pe n poziții binare în cod direct, cod invers și cod complementar.

3.7. Reprezentarea numerelor reale

Numerele reale se reprezintă în calculator sub formă fracționară prin intermediul reprezentării în virgulă fixă sau în virgulă mobilă (virgulă flotantă).

Reprezentarea în virgulă fixă. În acest caz se consideră că toate numerele au virgula plasată în aceeași poziție, chiar dacă acest lucru nu corespunde formei externe de reprezentare. Procesul de translatare din forma externă în forma internă și invers se realizează cu ajutorul unor coeficienți de scalare aleși în mod corespunzător de programator.

De obicei, se consideră că virgula este plasată imediat după poziția cifrei-semn, caz în care numerele sînt fracții pure (fig. 3.8).

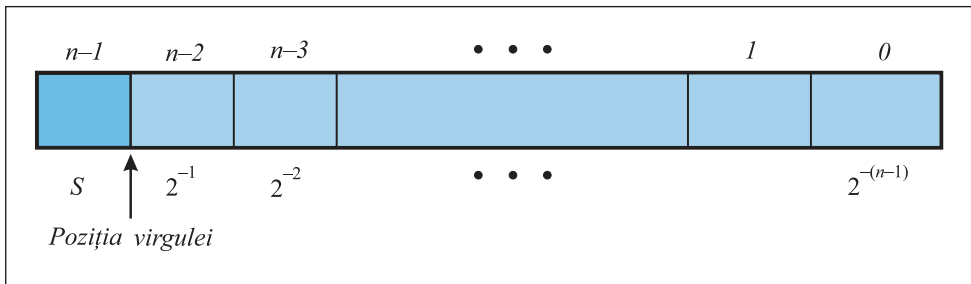


Fig. 3.8. Reprezentarea numerelor reale în virgulă fixă

Virgula însăși nu este materializată fizic în calculator. Din figura 3.8 rezultă că pe n poziții binare pot fi reprezentate numere reale, valoarea absolută a cărora este

$$0,00\dots 0 \leq |x| \leq 0,11\dots 1$$

sau, în sistemul zecimal,

$$0 \leq |x| \leq 1 - 2^{-(n-1)}.$$

Ca și în cazul numerelor întregi, numerele subunitare cu semn pot fi reprezentate, cu unele modificări, în cod direct, cod invers sau cod complementar.

Drept exemplu, în *figura 3.9* este redată reprezentarea numerelor

$$+0,9375 = +15/16 = (0,1111)_2,$$

$$-0,9375 = -15/16 = (-0,1111)_2$$

în virgulă fixă pe 8 poziții binare în cod direct.

7	6	5	4	3	2	1	0	
0	1	1	1	1	0	0	0	+0,9375
<i>S</i>	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	
7	6	5	4	3	2	1	0	
1	1	1	1	1	0	0	0	-0,9375
<i>S</i>	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	

Fig. 3.9. Reprezentarea numerelor +0,9375 și -0,9375 în virgulă fixă în cod direct

Avantajul principal al reprezentării în virgulă fixă constă în faptul că operațiile aritmetice cu numere reale pot fi efectuate de dispozitivul aritmetic destinat operării cu numere întregi.

Reprezentarea în virgulă mobilă. Operațiile în virgulă fixă sînt comode pentru circuitele omogene de date, cînd toate numerele reale sînt subunitare. Însă această reprezentare este inefficientă în calculele științifice, unde se lucrează simultan cu numere foarte mari și numere foarte mici al căror ordin de mărime adesea este imprevizibil. Pentru astfel de probleme se utilizează reprezentarea în virgulă mobilă.

Numerele reprezentate în virgulă mobilă pot fi numere întregi sau fracționare a căror valoare este dată de relația:

$$x = M \times b^E,$$

unde b este valoarea bazei, M este un număr subunitar numit **mantisă**, iar E este un **exponent**. În calculatoarele actuale se utilizează $b = 2$ sau 16.

Numărul reprezentat în virgulă mobilă este **normalizat** dacă prima cifră după virgulă a mantisei este diferită de zero.

Exemple:

1) Numărul 23 se va exprima în virgulă mobilă astfel:

$$23 = (10111)_2 = 0,10111 \times 2^5,$$

unde $M = 0,10111$, $b = 2$, $E = 5$.

2) Numărul 4,9375 se scrie în virgulă mobilă în felul următor:

$$4,9375 = (100,1111)_2 = 0,1001111 \times 2^3,$$

$$M = 0,1001111, b = 2, E = 3.$$

3) Numărul $-0,375$ se scrie în virgulă mobilă după cum urmează:

$$-0,375 = (-0,011)_2 = -0,11 \times 2^{-1},$$

$$M = -0,11, b = 2, E = -1.$$

Se observă că poziția reală a virgulei în cadrul numărului depinde de valoarea exponentului, adică virgula este mobilă (flotantă).

Există mai multe variante de reprezentare a mantisei și a exponentului pe n poziții binare. În figura 3.10 este redată reprezentarea în virgulă mobilă în **formatul exponent-mantisă**.

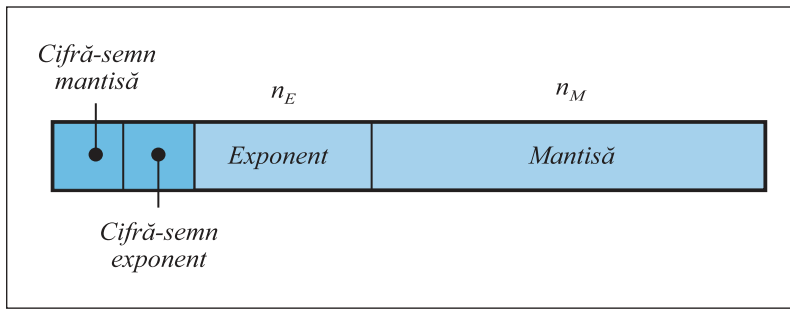


Fig. 3.10. Reprezentarea în virgulă mobilă, formatul exponent-mantisă

Numărul pozițiilor binare n_E alocate exponentului determină domeniul de mărime al numerelor care pot fi reprezentate, în timp ce numărul biților pentru mantisă n_M determină precizia de exprimare a numărului. În calculatoarele actuale $n_E = 6 \dots 15$ și $n_M = 24 \dots 64$, fapt ce permite reprezentarea numerelor într-un domeniu foarte larg al ordinului de mărime.

În figura 3.11 este redată reprezentarea numerelor în formatul **caracteristică-mantisă**.

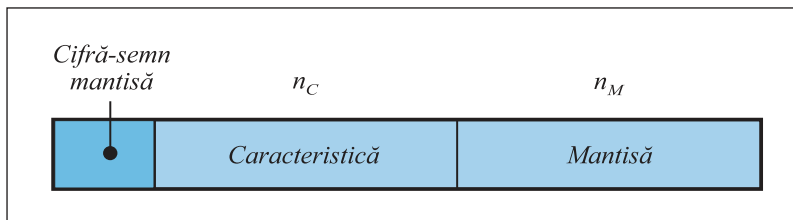


Fig. 3.11. Reprezentarea în virgulă mobilă, formatul caracteristică-mantisă

Caracteristica C constituie o formă de exprimare a exponentului E și se determină din relația

$$C = E + K.$$

În calculatoarele personale

$$K = 2^{n_c-1} - 1,$$

unde n_c este numărul de poziții binare alocate caracteristicii. În această reprezentare caracteristica, spre deosebire de exponent, nu poate lua valori negative, fapt ce simplifică structura dispozitivelor aritmetice.

În particular, pentru $n = 8$, caracteristica este calculată astfel:

$$C = E + 127,$$

indicînd practic semnul exponentului:

- dacă $C \geq 127$, atunci $E \geq 0$;
- dacă $C < 127$, atunci $E < 0$.

În *tabelul 3.5* sînt prezentate formatele caracteristică-mantisă, utilizate în majoritatea calculatoarelor personale.

Tabelul 3.5

Formatele caracteristică-mantisă în calculatoarele personale

Denumirea formatului	n	n_c	n_m	Precizia mantisei, cifre zecimale	Domeniul de mărimi al numerelor
Simplă precizie	32	8	23	6 sau 7	$10^{-37} \dots 10^{38}$
Dublă precizie	64	11	52	15 sau 16	$10^{-307} \dots 10^{308}$
Precizie extinsă	80	15	64	19	$10^{-4932} \dots 10^{4932}$

Întrucît conform condiției de normalizare, prima cifră a mantisei întotdeauna este cifra 1, acest bit poate să se reprezinte (să ocupe o poziție) sau nu în memoria calculatorului. În cazul în care nu se reprezintă, acest bit se numește **bit ascuns**. Accentuăm că tehnica bitului ascuns se referă doar la reprezentarea numerelor în memoria calculatorului, nu și la operațiile efectuate de dispozitivul aritmetic.

Întrebări și exerciții

- ❶ Care este deosebirea dintre reprezentarea în virgulă fixă și reprezentarea în virgulă mobilă?
- ❷ Reprezentați în virgula fixă pe 8 poziții binare în cod direct:

a) +0,875;	e) -0,875;	i) +0,125;
b) -0,125;	f) +0,3;	j) -0,3;
c) +0,4;	g) -0,4;	k) +0,15625;
d) -0,15625;	h) +0,21875;	l) -0,21875.
- ❸ Care sînt avantajele și dezavantajele reprezentării în virgulă fixă?
- ❹ Cum se reprezintă numerele reale în virgula mobilă? Care numere reprezentate în virgulă mobilă respectă condiția de normalizare?
- ❺ Cum se efectuează normalizarea numerelor reprezentate în virgulă mobilă?

⑥ Exprimați în virgulă mobilă următoarele numere:

a) +1,5;

e) -1,5;

i) -0,0625;

b) +0,0625;

f) +3,25;

j) +2,7;

c) -3,25;

g) -32,87;

k) -2,7;

d) -6,25;

h) +6,25;

l) -0,147.

⑦ De ce depinde precizia și domeniul de mărime ale numerelor reprezentate în virgulă mobilă?

⑧ Care este deosebirea dintre formatele exponent-mantisă și caracteristică-mantisă? Cum se calculează caracteristicile numerelor reprezentate în virgulă mobilă?

⑨ Calculați caracteristicile numerelor din exercițiul 6. Se consideră că $n_C = 8$.

⑩ Explicați termenul „bit ascuns”. Care sînt avantajele acestei reprezentări?

⑪ Cum se reprezintă numerele naturale, întregi și reale în calculatorul la care lucrăm dvs.? Determinați numărul de poziții binare alocate fiecărei reprezentări.

Test de autoevaluare nr. 3

1. Transformați în sistemul zecimal numerele ce urmează:

a) $(821)_9$;

b) $(1011,121)_3$;

c) $(341,52)_7$.

2. În care dintre sistemele de numerație a), b), c), d) scrierea numărului 284,6 este incorectă? Argumentați răspunsul.

a) zecimal;

c) binar;

b) octal;

d) hexazecimal.

3. Elaborați un program PASCAL care transformă în sistemul zecimal numerele scrise în baza b , $b < 10$. Baza b se citește de la tastatură.

4. Transformați în sistemul binar numerele ce urmează:

a) $(13,889)_{10}$;

b) $(73,542)_8$;

c) $(A8,74F)_{16}$.

5. Transformați în sistemul octal numerele ce urmează:

a) $(9758,93)_{10}$;

b) $(1011011011,01011101)_2$;

c) $(DA86,B1)_{16}$.

6. Transformați în sistemul hexazecimal numerele ce urmează:

a) $(9471,8362)_{10}$;

b) $(10110111011011,0101001101)_2$;

c) $(425,376)_8$.

7. Scrieți numerele $(1000001111)_2$, $(132)_8$, $(BB)_{16}$, $(222221)_4$ în ordine descrescătoare.

8. Printre numerele de mai jos există și numere egale. Scrieți egalitățile respective, de exemplu, $(1010)_2 = (12)_8$.

a) $(1010)_8$;

c) $(723)_8$;

e) $(21)_{10}$;

b) $(1D3)_{16}$;

d) $(25)_8$;

f) $(A)_{16}$.

9. Elaborați un program PASCAL pentru conversiunea octal-binară.

10. Calculați în sistemul binar:

a) $110001101 + 10111010;$ c) $111011 \times 101;$

b) $1101001010 - 101101001;$ d) $101011100 : 110.$

11. Reprezentați pe 8 poziții binare numărul natural 125.

12. Reprezentați pe 8 poziții binare în cod complementar numărul întreg -91 .

13. Cuvintele binare de mai jos reprezintă numere întregi scrise în cod invers pe 8 poziții binare. Scrieți aceste numere în sistemul zecimal.

a) $11111110;$ b) $00111111;$ c) $11011000.$

14. Reprezentați în virgula fixă pe 8 poziții binare în cod direct numărul $-0,75$.

15. Exprimați în virgulă mobilă următoarele numere:

a) $+21,125;$ b) $-73,25;$ c) $-0,09375.$

16. „Cutia neagră” transformă numerele zecimale în numere scrise în baza b . Determinați această bază și numărul x de la ieșirea din „cutia neagră” în cazul al doilea.



17. Cuvintele binare de mai jos reprezintă numere întregi scrise în cod complementar pe 8 poziții binare. Scrieți aceste numere în sistemul zecimal.

a) $11100110;$ b) $00101101;$ c) $11111000.$

18. Precizați intervalul numerelor întregi N care se pot reprezenta în cod complementar pe n poziții binare:

a) $n = 4;$ b) $n = 8;$ c) $n = 16.$

19. Cuvintele binare de mai jos reprezintă numere subunitare în virgulă fixă scrise în cod direct pe 4 poziții binare. Scrieți aceste numere în sistemul zecimal.

a) $1011;$ b) $0001;$ c) $1100.$

20. Scrieți în sistemul zecimal cel mai mic (A) și cel mai mare (B) numere ce pot fi reprezentate în virgulă fixă în cod direct pe 8 poziții binare.

21. Cuvintele binare de mai jos reprezintă numere reale în virgulă mobilă, baza $b = 2$, formatul exponent-mantisă, scrise în cod direct. Exponentul ocupă $n_E = 2$, iar mantisa $n_M = 4$ poziții binare. Scrieți aceste numere în sistemul zecimal.

a) $11111110;$ b) $00111111;$ c) $11011000.$

22. Reprezentați în virgulă mobilă, baza $b = 2$, formatul exponent-mantisă, în cod direct numărul $-2,75$. Exponentul ocupă $n_E = 2$, iar mantisa $n_M = 4$ poziții binare.

23. Scrieți în sistemul zecimal cel mai mic (A) și cel mai mare (B) numere ce pot fi reprezentate în virgulă mobilă, baza $b = 2$, formatul exponent-mantisă, cod direct. Exponentul ocupă $n_E = 2$, iar mantisa $n_M = 4$ poziții binare.

4.1. Variabile și expresii logice

Algebra booleană sau **algebra logică** este un compartiment al matematicii în care legile gândirii – obiectul de studiu al logicii clasice – sînt studiate cu ajutorul metodelor simbolice. Denumirea aceasta a fost dată în onoarea matematicianului englez George Boole, care în lucrarea *The Laws of Thought* („Legile gândirii”), publicată în 1853, a pus bazele acestei algebre. Mulți ani algebra booleană a fost considerată drept o simplă curiozitate matematică, fără a i se găsi o utilizare într-un anumit domeniu al științelor aplicate. Ea a revenit în actualitate odată cu apariția centralelor telefonice automate și a calculatoarelor numerice.

Din punct de vedere formal, algebra booleană poate fi definită printr-o mulțime a elementelor $\{0, 1\}$, o mulțime a operatorilor elementari $\{\neg, \&, \vee\}$ și printr-un set de postulate. Prin urmare, în algebra booleană orice variabilă nu poate avea decît una din cele două valori posibile, notate simbolic prin 0 și 1, alte valori neavînd nicio semnificație.

Variabilele algebrei booleene se notează prin $x, y, z, x_1, x_2, \dots$ cu sau fără indicii, iar elementele 0 și 1 se numesc **constante logice**.

Operatorii elementari ai algebrei booleene au următoarele denumiri:

- $\neg$ – negația (inversia, operația logică *NU*);
- $\&$ – conjuncția (produsul logic, operația logică *ȘI*);
- $\vee$ – disjuncția (suma logică, operația logică *SAU*).

Operatorii elementari se definesc cu ajutorul unor tabele speciale, numite **tabele de adevăr**.

Tabelul de adevăr este un tabel care include toate combinațiile posibile ale valorilor variabilelor față de care este definit operatorul și rezultatul operației respective.

În figura 4.1 sînt prezentate tabelele de adevăr ale negației, conjuncției și disjuncției.

x	$\bar{x}$
0	1
1	0

x	y	$x \& y$
0	0	0
0	1	0
1	0	0
1	1	1

x	y	$x \vee y$
0	0	0
0	1	1
1	0	1
1	1	1

Fig. 4.1. Tabelele de adevăr ale operatorilor elementari

Menționăm că în algebra booleană negația se indică printr-o linie orizontală amplasată deasupra variabilei respective.

Întrucât variabila x poate avea numai valoarea 0 sau 1, tabelul de adevăr al negației conține două rînduri. În cazurile conjuncției și disjuncției tabelele de adevăr conțin patru rînduri, cîte un rînd pentru fiecare din combinațiile posibile 00, 01, 10 și 11 ale variabilelor x și y .

Variabilele și constantele logice, reunite cu ajutorul operatorilor $\neg$, $\&$ și $\vee$ formează **expresii logice**, de exemplu:

$$1) \quad x \& y \vee z;$$

$$4) \quad 1 \& x \vee y;$$

$$2) \quad x \vee y \& z;$$

$$5) \quad 0 \vee x \vee y.$$

$$3) \quad \bar{x} \& y \vee z;$$

$$6) \quad 1 \vee 0.$$

Valorile expresiilor logice pot fi calculate utilizînd tabelele de adevăr ale operatorilor elementari. Pentru evaluarea expresiilor se stabilește următoarea **prioritate a operațiilor logice**:

- 1) negația;
- 2) conjuncția;
- 3) disjuncția.

De exemplu, în expresia

$$x \vee y \& z$$

mai întîi se execută operația $\&$, iar după aceea operația $\vee$. În expresia

$$x \& y \vee \bar{z}$$

se execută negația, în continuare conjuncția și, în sfîrșit, disjuncția.

Ordinea executării operațiilor logice poate fi schimbată cu ajutorul parantezelor „(” și „)”. Evident, în primul rînd, se execută operațiile dintre paranteze.

De exemplu, în cazul expresiei logice

$$x \vee y \& \bar{x} \vee z$$

se execută negația, conjuncția și disjuncțiile respective, iar în cazul expresiei

$$(x \vee y) \& (\bar{x} \vee z)$$

după negație se vor executa disjuncțiile și, pe urmă, conjuncția.

Pentru a sistematiza calculele, evaluarea expresiilor logice se efectuează în tabele speciale, numite **tabele de adevăr ale expresiilor logice**.

Tabelul de adevăr al expresiei logice include toate combinațiile posibile ale valorilor variabilelor din expresia examinată și rezultatele operațiilor logice în ordinea calculării lor.

De exemplu, în figura 4.2 este reprezentat tabelul de adevăr al expresiei,

$$\bar{x} \& y \vee z,$$

iar în figura 4.3 – tabelul de adevăr al expresiei

$$\overline{x \& y \vee z}.$$

x	y	z	$\bar{x}$	$\bar{x} \& y$	$\bar{x} \& y \vee z$
0	0	0	1	0	0
0	0	1	1	0	1
0	1	0	1	1	1
0	1	1	1	1	1
1	0	0	0	0	0
1	0	1	0	0	1
1	1	0	0	0	0
1	1	1	0	0	1

Fig. 4.2. Tabelul de adevăr al expresiei logice $\bar{x} \& y \vee z$

x	y	z	$x \& y$	$x \& y \vee z$	$\overline{x \& y \vee z}$
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	0	1	0
1	0	0	0	0	1
1	0	1	0	1	0
1	1	0	1	1	0
1	1	1	1	1	0

Fig. 4.3. Tabelul de adevăr al expresiei logice $\overline{x \& y \vee z}$

Menționăm că, pentru a simplifica notațiile, se admite ca simbolul operatorului $\&$ să fie omis din expresiile logice. De exemplu, expresiile logice

$$\bar{x} \& y \vee z,$$

$$(x \vee y) \& (\bar{x} \vee z)$$

pot fi scrise și în forma:

$$\bar{x}y \vee z,$$

$$(x \vee y)(\bar{x} \vee z).$$

Întrebări și exerciții

- ❶ Ce valori poate avea o variabilă logică?
- ❷ Care sînt operatorii elementari ai algebrei booleene și cum se definesc ei?
- ❸ Memorizați tabelele de adevăr ale negației, conjuncției și disjuncției.
- ❹ Cum se formează expresiile logice? Care este prioritatea operațiilor logice?
- ❺ Explicați rolul parantezelor din componența expresiilor logice.
- ❻ Alcătuiți tabelele de adevăr ale următoarelor expresii logice:

a) $\bar{x}y$;

c) $\bar{x} \vee y$;

b) $\bar{x}\bar{y}$;

d) $\overline{x \vee y}$;

- | | |
|----------------------|-----------------------------|
| e) $\bar{x};$ | j) $x \vee y \vee z;$ |
| f) $\bar{x}x;$ | k) $xy \vee z;$ |
| g) $\overline{xy};$ | l) $x (y \vee z);$ |
| h) $\bar{x} \vee x;$ | m) $\bar{x} \vee y \vee z;$ |
| i) $xyz;$ | n) $\bar{x} (y \vee z).$ |

- 7 Elaborati un program PASCAL care introduce valorile logice ale variabilelor x , y , z și afișează valorile uneia dintre expresiile ce urmează:

- | | |
|-------------------------------|----------------------------------|
| a) $\bar{x};$ | f) $x \vee y \vee z;$ |
| b) xy | g) $xy \vee z;$ |
| c) $x \vee y;$ | h) $\overline{x \vee y \vee z};$ |
| d) $\overline{(x \vee y)} z;$ | i) $x \vee xy;$ |
| e) $\overline{xy} \vee z;$ | j) $\bar{x} \vee y \vee z.$ |

- 8 Elaborati un program PASCAL care afișează tabelele de adevăr ale conjuncției și disjuncției.
- 9 Elaborati un algoritm repetitiv care formează toate combinațiile posibile ale valorilor variabilelor x , y și z . Afișați combinațiile respective pe ecran.
- 10 Pentru fiecare dintre expresiile logice ce urmează elaborati câte un program PASCAL care alcătuiește tabelul de adevăr respectiv:

- | | |
|---------------------------|----------------------------------|
| a) $x \vee \bar{x};$ | f) $\bar{x}y;$ |
| b) $x\bar{x};$ | g) $x \vee y \vee z;$ |
| c) $\overline{x \vee y};$ | h) $xyz;$ |
| d) $\bar{x}\bar{y};$ | i) $(x \vee y)(\bar{x} \vee y);$ |
| e) $\bar{x} \vee y;$ | j) $(x \vee y)(x \vee \bar{y}).$ |

4.2. Funcții logice

Noțiunea de **funcție logică** sau **funcție booleană** se definește în același mod ca și în cazul algebrei clasice.

Vom nota prin $x_1, x_2, \dots, x_n$ un grup arbitrar de variabile booleene, unde $n = 1, 2, 3, \dots$. Întrucât fiecare variabilă booleană poate avea numai valorile 0 sau 1, numărul tuturor combinațiilor posibile ale valorilor variabilelor $x_1, x_2, \dots, x_n$ este de 2^n .

Firește, pentru $n = 1$ avem 2 combinații (0 și 1); pentru $n = 2$ sînt $2^2 = 4$ combinații (00, 01, 10 și 11); pentru $n = 3$ există $2^3 = 8$ combinații (000, 001, 010, 011, 100, 101, 110 și 111) etc.

Funcția logică de n variabile $y = f(x_1, x_2, \dots, x_n)$ este o aplicație care pune în corespondență fiecărei combinații de valori ale variabilelor $x_1, x_2, \dots, x_n$ valoarea 0 sau 1 a variabilei y .

Variabilele $x_1, x_2, \dots, x_n$ se numesc **variabile independente** sau **argumente**, iar variabila y – **variabilă dependentă** sau **funcție** de argumente $x_1, x_2, \dots, x_n$.

Prin urmare, **domeniul de definiție** al funcției $y = f(x_1, x_2, \dots, x_n)$ este mulțimea tuturor combinațiilor posibile

0 0 ... 0
0 0 ... 1
...
1 1 ... 1

ale valorilor argumentelor $x_1, x_2, \dots, x_n$, în total 2^n combinații, iar **domeniul valorilor** funcției logice este mulțimea $\{0, 1\}$.

Ca și în cazul algebrei clasice, funcțiile logice pot fi definite prin **tabele, formule și metode grafice**.

Tabelul de adevăr al funcției logice $y = f(x_1, x_2, \dots, x_n)$ este un tabel care include toate combinațiile posibile ale valorilor argumentelor $x_1, x_2, \dots, x_n$ și valorile corespunzătoare ale variabilei dependente y .

De exemplu, în *figura 4.4* este prezentat tabelul de adevăr al unei funcții logice de 3 variabile.

x_1	x_2	x_3	y
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

Fig. 4.4. Tabelul de adevăr al unei funcții logice de trei variabile

Tabelul include $2^3 = 8$ rânduri și are 2 coloane: prima pentru combinațiile posibile ale valorilor argumentelor x_1, x_2, x_3 și a doua pentru valorile corespunzătoare ale variabilei dependente y . Conform tabelului analizat, combinației $x_1 = 0, x_2 = 0, x_3 = 0$ îi corespunde valoarea $y = f(0,0,0) = 1$; combinației $x_1 = 0, x_2 = 0, x_3 = 1$ – valoarea $y = f(0,0,1) = 0$ etc.

Definirea funcției logice prin formule se face atribuind variabilei dependente y valorile expresiilor logice ce conțin argumentele $x_1, x_2, \dots, x_n$.

De exemplu,

1) $y = x_1$;

2) $y = x_1 x_2$;

3) $y = \bar{x}_1 x_2 \vee x_3$;

4) $y = \bar{x}_1 x_2 \vee \bar{x}_3$.

Natural, cunoscînd formula unei funcții logice, poate fi calculat tabelul ei de adevăr. De exemplu, tabelul de adevăr al funcției

$$y = \overline{x_1 x_2} \vee x_3$$

va fi cel prezentat în *figura 4.4*. Ca să ne convingem de acest fapt, e suficient să calculăm tabelul de adevăr al expresiei logice

$$\overline{x_1 x_2} \vee x_3$$

prezentat, prin alte notații, în *figura 4.3* din paragraful 4.1.

Există mai multe **metode grafice de definire a funcțiilor logice**. Aceste metode se bazează pe diagramele utilizate în teoria mulțimilor și se studiază în cursurile avansate de informatică.

Întrebări și exerciții

- ❶ Formulați definiția noțiunii „funcție logică”.
- ❷ Care sînt domeniul de definiție și domeniul de valori ale unei funcții logice?
- ❸ Prin ce metode poate fi definită o funcție logică de n variabile?
- ❹ Cum se alcătuește tabelul de adevăr al unei funcții de n variabile? Cîte rînduri conține acest tabel?
- ❺ Cum poate fi alcătuit tabelul de adevăr al unei funcții logice atunci cînd se cunoaște formula ei?
- ❻ Funcția logică de 3 variabile $y = f(x_1, x_2, x_3)$ este definită prin tabelul de adevăr din *figura 4.4*. Numiți combinațiile valorilor argumentelor x_1, x_2, x_3 pentru care funcția dată are valoarea $y = 1$. Numiți combinațiile respective pentru valoarea funcției $y = 0$.
- ❼ Alcătuiți tabelele de adevăr ale următoarelor funcții logice:

a) $y = x;$

f) $y = \overline{x_1 \vee x_2};$

b) $y = \bar{x};$

g) $y = \overline{x_1 x_2 x_3};$

c) $y = x_1 x_2;$

h) $y = x_1 (x_2 \vee \bar{x}_3);$

d) $y = x_1 \vee x_2;$

i) $y = \bar{x}_1 \vee x_2 x_3;$

e) $y = \overline{x_1 x_2};$

j) $y = x_1 \vee x_2 \overline{x_3};$

- ❽ Elaborați un program PASCAL care introduce valorile logice ale variabilelor x_1, x_2, x_3, x_4 și afișează valorile uneia dintre funcțiile ce urmează:

a) $y = x_1 x_2 \vee x_3 x_4;$

g) $y = x_1 \bar{x}_2 \vee x_3 \bar{x}_4;$

b) $y = (x_1 \vee x_2)(x_3 \vee x_4);$

h) $y = \bar{x}_1 x_2 \vee \bar{x}_3 x_4;$

c) $y = \overline{x_1 x_2 x_3 x_4};$

i) $y = \bar{x}_1 \bar{x}_2 \vee \bar{x}_3 \bar{x}_4;$

d) $y = \overline{x_1 \vee x_2 \vee x_3 \vee x_4};$

j) $y = x_1 x_2 x_3 x_4 \vee \bar{x}_2 \bar{x}_3 x_4;$

e) $y = \overline{x_1 x_2} \vee \overline{x_3 x_4};$

k) $y = x_1 \vee x_2 x_3 \vee x_4;$

f) $y = \overline{(x_1 \vee x_2)(x_3 \vee x_4)};$

l) $y = x_1 \vee x_2 \vee x_3 \vee \bar{x}_1 \bar{x}_2 \bar{x}_3 \bar{x}_4.$

9 Elaborati un program PASCAL care alcătuiește tabelul de adevăr al uneia dintre funcțiile ce urmează:

- | | |
|------------------------------------|---|
| a) $y = x;$ | h) $y = \bar{x}_1 \bar{x}_2;$ |
| b) $y = \bar{x};$ | i) $y = \bar{x}_1 x_2 \bar{x}_3;$ |
| c) $y = x_1 x_2;$ | j) $y = \bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3;$ |
| d) $y = x_1 \vee x_2;$ | k) $y = \overline{x_1 \vee x_2 \vee x_3};$ |
| e) $y = \overline{x_1 x_2};$ | l) $y = \bar{x}_1 \bar{x}_2 \bar{x}_3;$ |
| f) $y = \bar{x}_1 \vee \bar{x}_2;$ | m) $y = x_1 \vee x_2 \vee \bar{x}_1 \bar{x}_2 \bar{x}_3 \bar{x}_4;$ |
| g) $y = \overline{x_1 \vee x_2};$ | n) $y = x_1(x_2 \vee \bar{x}_3 \vee \bar{x}_4).$ |

4.3. Funcții logice frecvent utilizate

Să admitem n variabile independente $x_1, x_2, \dots, x_n$. Apare întrebarea, câte funcții logice de n variabile există în algebra booleană? **Numărul funcțiilor logice posibile** poate fi determinat prin următoarele raționamente.

Întrucât orice funcție logică poate fi definită cu ajutorul tabelului de adevăr, numărul funcțiilor posibile de n variabile coincide cu numărul tabelelor distincte de adevăr.

Pentru a defini o funcție logică, în coloana y a tabelului de adevăr se indică valorile funcției – 0 sau 1 pentru fiecare dintre cele 2^n combinații ale valorilor argumentelor. Întrucât tabelul de adevăr are 2^n rînduri, există

$$m = 2^{2^n}$$

funcții logice de n variabile. Funcțiile respective se notează prin $y_j, j = 0, 1, \dots, m - 1$.

De exemplu, în cazul în care $n = 1$, există $m = 2^{2^1} = 2^2 = 4$ funcții logice, reprezentate în figura 4.5.

x	y_0	y_1	y_2	y_3
0	0	1	0	1
1	0	0	1	1

Fig. 4.5. Funcții logice de o singură variabilă

Evident,

$$\begin{aligned}
 y_0 &= f(x) = 0; \\
 y_1 &= f(x) = \bar{x}; \\
 y_2 &= f(x) = x; \\
 y_3 &= f(x) = 1.
 \end{aligned}$$

Funcțiile y_0 și y_3 se numesc **funcția constanta 0** și, respectiv, **constantă 1**. Funcția y_1 este **funcția logică NU** sau **negația**, iar funcția y_2 se numește **funcția de repetare**.

Pentru $n = 2$ există

$$m = 2^{2^2} = 2^4 = 16$$

funcții logice, reprezentate în figura 4.6.

x_1	x_2	y_0	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9	y_{10}	y_{11}	y_{12}	y_{13}	y_{14}	y_{15}
0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
0	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	0	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Fig. 4.6. Funcții logice de două variabile

Funcțiile y_0 și y_{15} sînt funcțiile deja cunoscute, constanta 0 și, respectiv, constanta 1:

$$y_0 = f(x_1, x_2) = 0;$$

$$y_{15} = f(x_1, x_2) = 1.$$

Funcția y_8 poate fi scrisă în forma:

$$y_8 = f(x_1, x_2) = x_1 x_2.$$

În mod firesc, funcția y_8 va avea denumirea **funcția logică ȘI** sau **conjuncția**.

Funcția y_{14} poate fi scrisă în forma:

$$y_{14} = f(x_1, x_2) = x_1 \vee x_2.$$

Prin urmare, funcția y_{14} va avea denumirea **funcția logică SAU** sau **disjuncția**.

Funcțiile logice NU, ȘI, SAU, induse de operatorii elementari, respectiv, $\neg$, $\&$, $\vee$ se numesc funcții logice elementare.

Din figura 4.6 se observă că

$$y_1 = \overline{x_1 \vee x_2}.$$

Funcția dată se numește **funcția logică SAU-NU**.

În mod similar, funcția y_7 poate fi exprimată în forma

$$y_7 = f(x_1, x_2) = \overline{x_1 x_2}.$$

Funcția în cauză se numește **funcția logică ȘI-NU**.

Funcția

$$y_9 = f(x_1, x_2) = \overline{x_1} \overline{x_2} \vee x_1 x_2$$

are valoarea 1 numai cînd $x_1 = x_2 = 0$ sau $x_1 = x_2 = 1$. Această funcție se numește **funcția logică COINCIDENȚĂ** sau **echivalență** și se notează prin simbolul „ $\equiv$ ”.

Analizînd tabelul din figura 4.6, mai observăm că

$$y_3 = f(x_1, x_2) = \overline{x_1} \text{ (negația variabilei } x_1);$$

$$y_{12} = f(x_1, x_2) = x_1 \text{ (repetarea variabilei } x_1);$$

$$y_5 = f(x_1, x_2) = \overline{x_2} \text{ (negația variabilei } x_2);$$

$$y_{10} = f(x_1, x_2) = x_2 \text{ (repetarea variabilei } x_2).$$

Pentru a fi mai ușor memorizate, în figura 4.7 sînt prezentate tabelele de adevăr ale funcțiilor logice **NU**, **ȘI**, **SAU**, **ȘI-NU**, **SAU-NU** și **COINCIDENȚĂ**.

NU		ȘI			SAU		
x	$\bar{x}$	x_1	x_2	$x_1 x_2$	x_1	x_2	$x_1 \vee x_2$
0	1	0	0	0	0	0	0
1	0	0	1	0	0	1	1
		1	0	0	1	0	1
		1	1	1	1	1	1

ȘI-NU			SAU-NU			COINCIDENTĂ		
x_1	x_2	$\overline{x_1 x_2}$	x_1	x_2	$\overline{x_1 \vee x_2}$	x_1	x_2	$x_1 \equiv x_2$
0	0	1	0	0	1	0	0	1
0	1	1	0	1	0	0	1	0
1	0	1	1	0	0	1	0	0
1	1	0	1	1	0	1	1	1

Fig. 4.7. Funcțiile logice frecvent utilizate

În mod similar pot fi studiate funcțiile logice de 3 variabile, numărul cărora este de $m = 2^3 = 2^8 = 256$; funcțiile de 4 variabile, numărul cărora este de $m = 2^4 = 2^{16} = 65\,536$ etc. Se observă că, deși este finit, numărul funcțiilor booleene posibile este enorm. S-a demonstrat însă că **orice funcție logică de n variabile, $n \geq 2$, poate fi exprimată printr-o formulă care include numai operatorii elementari $\bar{}$, $\&$, $\vee$** . Această proprietate facilitează realizarea tehnică a dispozitivelor destinate calculării funcțiilor logice cu un număr arbitrar de argumente.

Întrebări și exerciții

- 1 Determinați numărul funcțiilor logice de 5 și de 6 variabile.
- 2 Numiți funcțiile logice elementare și alcătuiți tabelele respective de adevăr.
- 3 Memorizați tabelele de adevăr ale funcțiilor logice frecvent utilizate *NU*, *ȘI*, *SAU*, *ȘI-NU*, *SAU-NU* și *COINCIDENTĂ*.
- 4 Elaborați un program care va afișa pe ecran tabelul de adevăr al uneia dintre funcțiile logice y_0, y_1, y_2, y_3 de 2 variabile.
- 5 Elaborați un program care va afișa pe ecran tabelul de adevăr al uneia dintre funcțiile logice y_j de n variabile.

Test de autoevaluare nr. 4

1. Alcătuiți tabelul de adevăr al expresiei logice $x \vee y\bar{z}$.

2. Care dintre expresiile logice ce urmează sînt egale? Amintim că două expresii logice sînt egale dacă valorile lor coincid pentru toate combinațiile posibile ale valorilor variabilelor respective.

a) $(\bar{x} \vee \bar{y})(x \vee y)$;

c) $(x \vee y) \vee x\bar{y}$;

b) $xy \vee x\bar{y}$;

d) $x\bar{y} \vee \bar{x}y$.

3. Scrieți toate combinațiile posibile de valori ale variabilelor x, y, z pentru care valoarea expresiei $xy \vee \bar{z}$ este egală cu 1.

4. Elaborați un program PASCAL care citește de la tastatură valorile variabilelor logice x, y, z și afișează pe ecran valoarea expresiei $x\bar{y} \vee z$.

5. Elaborați un program PASCAL care afișează pe ecran tabelul de adevăr al expresiei $\bar{x}\bar{y} \vee z$.

6. Indicați domeniul de definiție și domeniul de valori ale funcției logice $y = x_1(x_2 \vee \bar{x}_3)$.

7. Alcătuiți tabelul de adevăr al funcției logice $y = x_1(\bar{x}_2 \vee x_3)$.

8. Elaborați un program PASCAL care citește de la tastatură valorile variabilelor logice x_1, x_2, x_3 și afișează pe ecran valoarea funcției $y = x_1(\bar{x}_2 \vee \bar{x}_3)$.

9. Elaborați un program PASCAL care afișează pe ecran tabelul de adevăr al funcției logice $y = \bar{x}_1(x_2 \vee x_3)$.

10. Determinați numărul funcțiilor logice de 5 variabile.

11. Alcătuiți tabelele de adevăr ale funcțiilor logice ȘI-NU și SAU-NU.

12. Care dintre funcțiile logice ce urmează sînt egale? Amintim că două funcții logice sînt egale dacă valorile lor coincid pentru toate combinațiile posibile ale valorilor variabilelor independente.

a) $y = \overline{x_1 \vee x_2};$

c) $y = \bar{x}_1\bar{x}_2;$

b) $y = \bar{x}_1x_2;$

d) $y = \bar{x}_1 \vee \bar{x}_2.$

13. Funcția logică de trei variabile $y = f(x_1, x_2, x_3)$ este definită cu ajutorul următorului tabel de adevăr:

x_1	x_2	x_3	y
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

Selectați din expresiile ce urmează formula care, de asemenea, definește această funcție:

a) $y = x_1 \vee x_2 \vee x_3;$

c) $y = \bar{x}_1x_2x_3;$

b) $y = \bar{x}_1 \vee x_2 \vee x_3;$

d) $y = \bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3.$

5.1. Circuite logice elementare

Circuitul logic este un dispozitiv destinat calculării funcțiilor logice. Pentru a realiza circuitele logice, e necesar ca valorile binare **0** și **1** ale argumentelor și funcțiilor respective să fie reprezentate prin valorile unor mărimi fizice, de exemplu: presiune, temperatură, tensiune sau curent electric, flux luminos etc. În funcție de mărimile fizice utilizate, deosebim dispozitive logice mecanice, pneumatice, hidraulice, electromecanice, electronice, optice etc. În dispozitivele hidraulice și pneumatice valorile logice **0** sau **1** pot fi reprezentate prin valorile mici și, respectiv, mari ale presiunii fluidului, în dispozitivele electromecanice și electronice – prin prezența sau absența curentului electric, prin niveluri de tensiune etc.

Pentru o înțelegere clară a principiilor de funcționare a dispozitivelor logice, vom studia mai întâi circuitele cu contacte. Componentele de bază ale acestor circuite sînt **elementele de comutare** – contactele electrice normal deschise și contactele electrice normal închise.

În cazul **contactelor normal deschise**, circuitul electric este deschis dacă contactele nu sînt acționate și – închis la acționarea lor. În cazul **contactelor normal închise**, circuitul electric este închis dacă contactele nu sînt acționate și – deschis la acționarea lor (fig. 5.1).

	<i>Neacționate</i>	<i>Acționate</i>
Contacte normal deschise		
Contacte normal închise		

Fig. 5.1. Contacte normal deschise și normal închise

De exemplu, contactele unui întrerupător electric uzual sînt contacte normal deschise, iar contactele butonului de pauză al magnetofonului sînt contacte normal închise.

În circuitele cu contacte, valorile logice ale argumentelor sînt reprezentate prin stările contactelor electrice respective. Valorii logice **1** îi corespunde starea „contactul este acționat”, iar valorii logice **0** îi corespunde starea „contactul este neacționat”.

Circuitul electric care realizează **funcția logică NU** și simbolul utilizat sînt prezentate în *figura 5.2*.

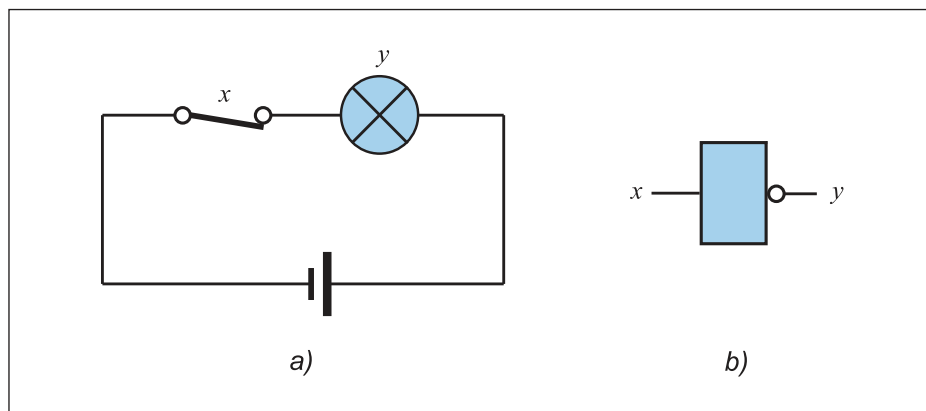


Fig. 5.2. Circuit cu contacte pentru realizarea funcției logice *NU* (a) și simbolul utilizat (b)

Argumentul x este materializat prin contactul normal închis, iar valorile variabilei dependente y sînt reprezentate prin stările becului electric: *stins* (valoarea logică **0**) sau *aprins* (valoarea logică **1**). Se observă că becul va fi aprins ($y=1$), dacă contactul normal închis nu este acționat ($x=0$).

Funcția logică ȘI se realizează prin conectarea în serie a contactelor electrice. Circuitul electric care realizează funcția *ȘI* de două variabile și simbolul utilizat sînt prezentate în *figura 5.3*.

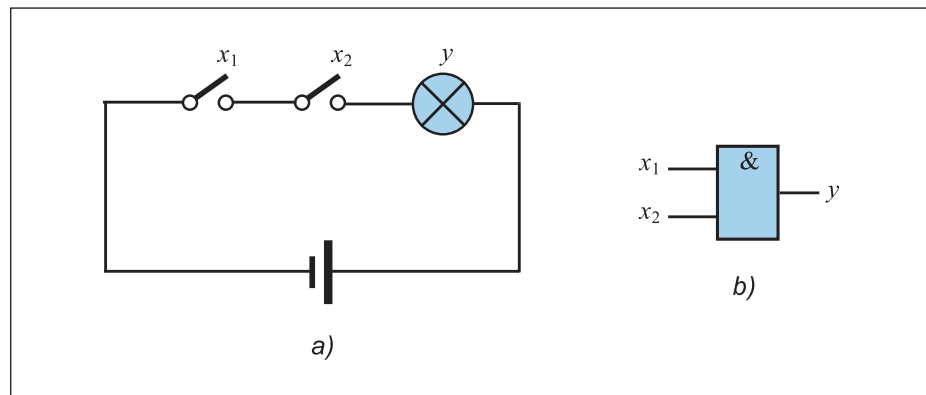


Fig. 5.3. Circuit cu contacte pentru realizarea funcției logice *ȘI* (a) și simbolul utilizat (b)

Variabilele x_1 și x_2 sînt materializate prin cele două contacte normale deschise, iar valorile variabilei y prin bec. Se observă că becul va fi aprins ($y=1$), numai dacă ambele contacte normale deschise sînt acționate ($x_1=1$ și $x_2=1$).

Funcția logică SAU se realizează prin conectarea în paralel a contactelor electrice. Circuitul respectiv este prezentat în *figura 5.4*.

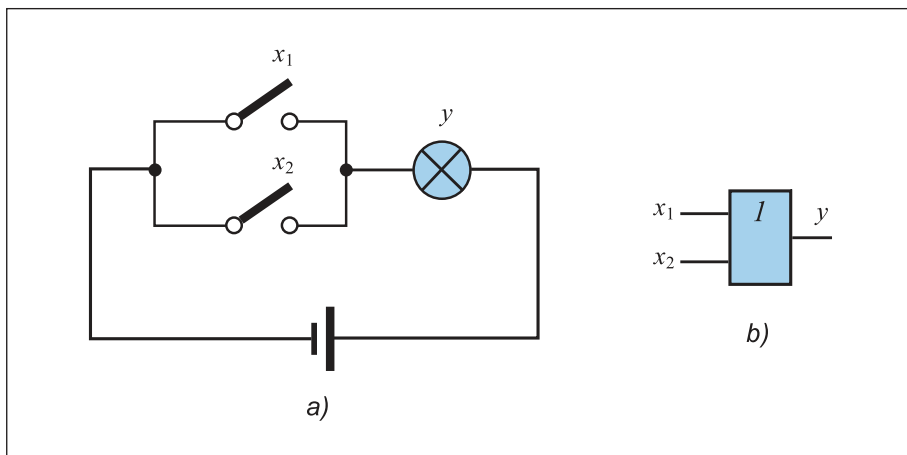


Fig. 5.4. Circuite cu contacte pentru realizarea funcției logice SAU (a) și simbolul utilizat (b)

Se constată că becul va fi aprins ($y=1$), dacă cel puțin unul dintre cele două contacte normale deschise este acționat ($x_1=1$ sau $x_2=1$).

Întrucât viteza de închidere-deschidere a contactelor electrice este foarte mică, în calculatoarele moderne valorile **0**, **1** sînt reprezentate prin niveluri de tensiune, iar ca element de comutare se utilizează tranzistorul.

Tranzistorul este un dispozitiv electronic format în sau pe suprafața unui monocristal semiconductor. Tehnologiile avansate permit fabricarea a $10^6 - 10^7$ de tranzistoare pe o suprafață de 1 cm^2 al monocristalului.

În regim de comutație, tranzistorul poate fi considerat ca un întrerupător obișnuit, care într-o stare conduce curentul (este închis), iar în alta – nu (este deschis). Spre deosebire însă de întrerupătorul obișnuit, închiderea sau deschiderea tranzistorului se realizează cu ajutorul curentului electric.

Există mai multe tipuri de tranzistoare. În figura 5.5 este prezentat tranzistorul *n-p-n* (abrevierea se referă la structura internă a tranzistorului) și schemele echivalente în regim de comutație.

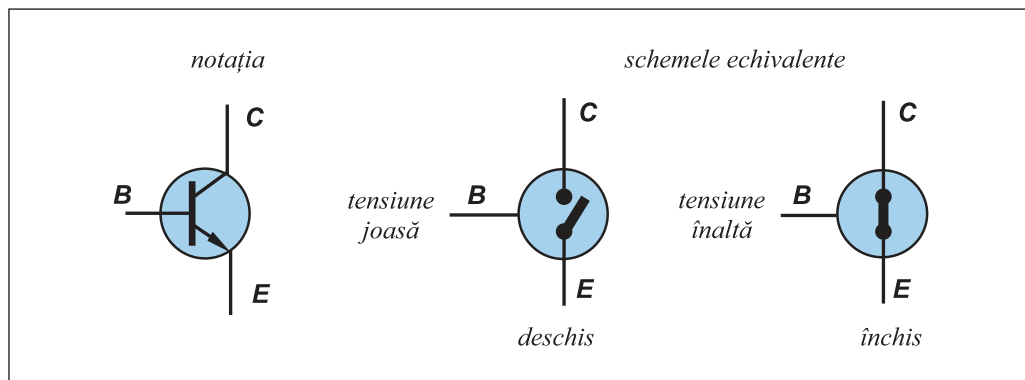


Fig. 5.5. Tranzistorul *n-p-n*

Tranzistorul $n-p-n$ are trei terminale: emitorul E , baza B și colectorul C . În regim de comutație, emitorul și colectorul pot fi considerați drept contacte care se închid sau se deschid cu ajutorul unei tensiuni aplicate la bază. Menționăm că tranzistoarele moderne permit efectuarea a $10^6 - 10^9$ închideri–deschideri pe secundă. Ca și în cazul contactelor electrice studiate mai sus, utilizarea diferitor tipuri de tranzistoare și conectarea lor în serie sau în paralel permite realizarea funcțiilor logice NU , ȘI , SAU .

Circuitele destinate calculării funcțiilor logice frecvent utilizate se numesc circuite logice elementare sau porți logice.

Simbolurile utilizate pentru notarea porților logice sînt reprezentate în figura 5.6.

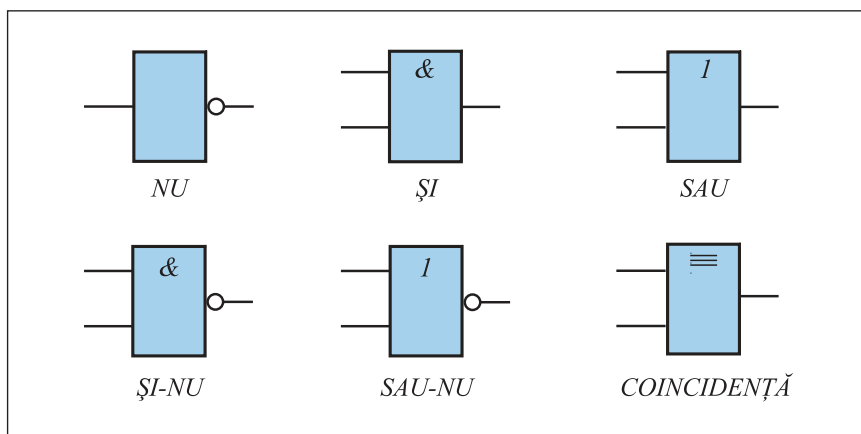


Fig. 5.6. Simbolurile porților logice

E cunoscut faptul că orice funcție logică poate fi exprimată printr-o formulă care include numai simbolurile operatorilor elementari $\neg$, $\&$, $\vee$. Prin urmare, **orice funcție logică cu un număr arbitrar de argumente poate fi materializată prin conectarea porților logice NU , ȘI , SAU** . De exemplu, funcția

$$y = x_1x_2 \vee \bar{x}_2x_3$$

poate fi realizată cu ajutorul următoarelor porți logice:

- o poartă NU pentru calcularea $\bar{x}_2$;
- două porți logice ȘI pentru calcularea conjuncțiilor x_1x_2 și $\bar{x}_2x_3$;
- o poartă SAU pentru calcularea disjuncției $x_1x_2 \vee \bar{x}_2x_3$;

Schema circuitului logic pentru calcularea funcției respective este prezentată în figura 5.7.

Întrebări și exerciții

- ❶ Cum pot fi reprezentate valorile binare **0** și **1**?
- ❷ Cum funcționează contactele normal deschise și contactele normal închise?
- ❸ Care este reprezentarea valorilor binare **0** și **1** în circuitele cu contacte?

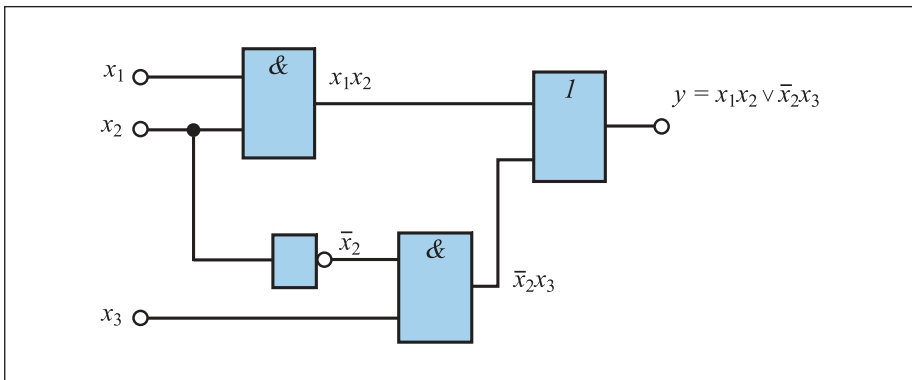


Fig. 5.7. Circuitul logic pentru realizarea funcției $y = x_1x_2 \vee \bar{x}_2x_3$

- 4 Utilizând trusa din laboratorul de fizică, montați circuitele din figurile 5.2, 5.3 și 5.4. Verificați tabelele de adevăr ale funcțiilor realizate de circuitele în cauză.
- 5 Cum se realizează funcțiile *NU*, *ȘI*, *SAU* în cazul circuitelor cu contacte electrice?
- 6 Care este rolul elementelor de comutare la realizarea circuitelor logice?
- 7 Care este rolul tranzistorului în calculatoarele moderne?
- 8 Memorizați simbolurile porților logice. Explicați cum se utilizează porțile logice la realizarea funcțiilor logice arbitrare.
- 9 Utilizând porțile *NU*, *ȘI*, *SAU*, elaborați circuitele logice pentru calcularea următoarelor funcții:

- | | |
|---|--|
| a) $y = x_1x_2x_3;$ | i) $y = x_1x_2 \vee \bar{x}_2\bar{x}_3;$ |
| b) $y = x_1 \vee x_2 \vee x_3;$ | j) $y = (x_1 \vee x_2) (\overline{x_2 \vee x_3});$ |
| c) $y = \bar{x}_1x_2x_3;$ | k) $y = x_1x_2 \vee \bar{x}_1x_3 \vee x_3x_4;$ |
| d) $y = \bar{x}_1 \vee x_2 \vee x_3;$ | l) $y = \bar{x}_1x_2 \vee \bar{x}_1x_3 \vee x_2x_3;$ |
| e) $y = x_1x_2 \vee x_3x_4;$ | m) $y = \bar{x}_1x_2 \vee x_1\bar{x}_2;$ |
| f) $y = (x_1 \vee x_2) (x_3 \vee x_4);$ | n) $y = x_1x_2 \vee \bar{x}_1\bar{x}_2;$ |
| g) $y = \bar{x}_1\bar{x}_2;$ | o) $y = x_1(x_2 \vee x_3 \vee x_4);$ |
| h) $y = \bar{x}_1 \vee \bar{x}_2;$ | p) $y = x_1 \vee \bar{x}_2\bar{x}_3\bar{x}_4.$ |

- 10 Releul electromagnetic este un dispozitiv cu care se comandă închiderea sau deschiderea contactelor electrice. Contactele respective sînt acționate de un electromagnet. Cum poate fi utilizat releul pentru realizarea funcțiilor logice *NU*, *ȘI*, *SAU*?

Cu piesele din trusa de fizică asamblați un releu electromagnetic și verificați soluțiile propuse de dvs.

- 11 Reprezentînd valorile binare ale variabilelor de ieșire prin prezența (valoarea 1) sau absența (valoarea 0) a fluidului, elaborați circuitele hidraulice pentru realizarea funcțiilor logice *NU*, *ȘI*, *SAU*. Montați instalațiile respective utilizînd robinetele și furtunurile din trusa de chimie. Verificați tabelele de adevăr ale funcțiilor logice realizate.

5.2. Clasificarea circuitelor logice

Circuitele logice se clasifică în două categorii: circuite combinaționale și circuite secvențiale.

Într-un **circuit combinațional** valorile variabilelor de ieșire sînt determinate în orice moment de combinația valorilor variabilelor de intrare conform funcțiilor logice ale circuitului. Într-un **circuit secvențial** valorile variabilelor de ieșire depind nu numai de combinațiile valorilor variabilelor de intrare, dar și de consecutivitatea aplicării lor. Altfel spus, circuitele combinaționale reprezintă circuite logice lipsite de elemente de memorie, iar circuitele secvențiale includ și elementele de memorie binară. Prin urmare, un circuit combinațional realizează o prelucrare numerică a informației, care poate fi în întregime exprimată printr-un grup de funcții logice în care nu intervine parametrul timp.

Schema-bloc a unui circuit combinațional este prezentată în figura 5.8.

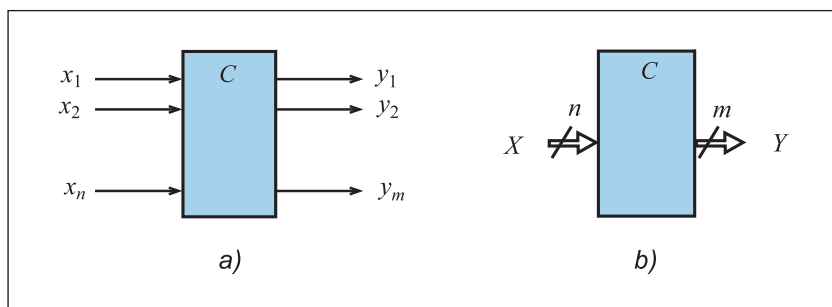


Fig. 5.8. Schema-bloc a unui circuit logic combinațional:
a – detaliat; b – generalizat

Circuitul are n variabile de intrare: $X = \langle x_1, x_2, \dots, x_n \rangle$, și m variabile de ieșire: $Y = \langle y_1, y_2, \dots, y_m \rangle$. Conexiunile destinate transmiterii valorilor unui grup de variabile se reprezintă prin linii duble. Dacă e necesar, numărul de variabile se indică lângă o bară care întretaie linia dublă a grupului de variabile. De exemplu, în figura 5.8 b se indică că grupul X include n variabile, iar grupul Y – m variabile.

Din punctul de vedere al teoriei informației, circuitul combinațional reprezintă un convertor de cod: la intrările X se aplică combinațiile admise de primul cod, din care se face conversiunea, iar la ieșirile Y apar combinațiile corespunzătoare în cel de al doilea cod, în care se face conversiunea.

De exemplu, primul cod poate fi codul *EBCDIC*, iar al doilea codul *ASCII*.

5.3. Sumatorul

Una dintre principalele sarcini ale unui calculator în momentul prelucrării informației constă în efectuarea operațiilor aritmetice elementare și, în special, adunarea și scăderea. Dispozitivele în care au loc aceste operații au la bază circuite cu ajutorul cărora se efectuează adunarea, respectiv scăderea a două cifre binare.

Semisumatorul este un circuit combinațional destinat adunării a două cifre binare. Tabelul de adevăr care pune în evidență funcționarea unui semisumator rezultă din regula de adunare a două cifre binare și este prezentat în *figura 5.9*. Aici a și b reprezintă cele două cifre binare care se adună, s – cifra-sumă a rangului respectiv, iar t – cifra de transport către rangul următor.

a	b	s	t
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Fig. 5.9. Tabelul de adevăr pentru adunarea a două cifre binare

Pentru a elabora o schemă posibilă a semisumatorului, exprimăm funcțiile de ieșire s și t :

$$s = \bar{a}b \vee a\bar{b};$$

$$t = ab.$$

Schema care realizează funcțiile s , t și simbolul utilizat sînt prezentate în *figura 5.10*.

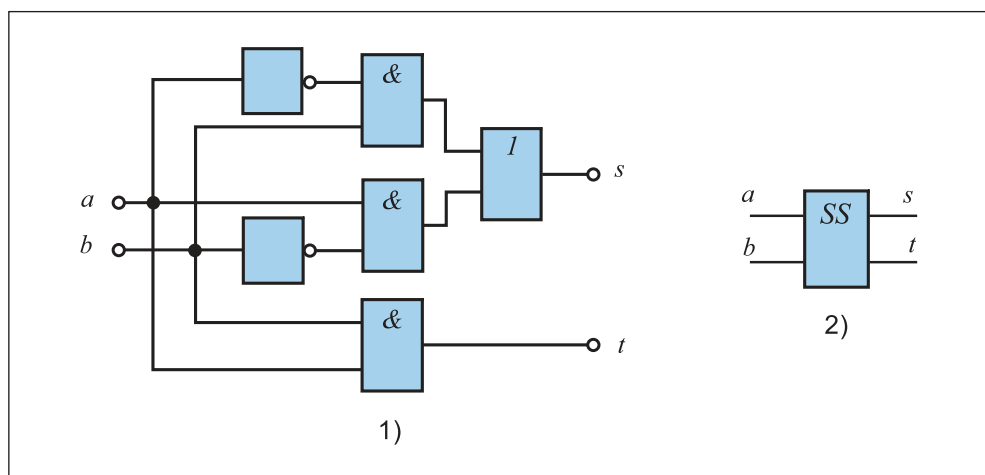


Fig. 5.10. Schema semisumatorului (1) și simbolul utilizat (2)

Să luăm două numere binare

$$A = a_{n-1} a_{n-2} \dots a_j \dots a_0$$

și

$$B = b_{n-1} b_{n-2} \dots b_j \dots b_0,$$

unde a_j și b_j reprezintă cifrele binare din rangul (poziția) j . La însumarea cifrelor a_j și b_j din rangul j trebuie să se ia în considerare și cifra de transport t_{j-1} de la rangul inferior $j-1$:

$$\begin{array}{c}
 t_{j-1} \\
 a_{n-1} \ a_{n-2} \ \dots \ a_j \ \dots \ a_0 \\
 + \\
 b_{n-1} \ b_{n-2} \ \dots \ b_j \ \dots \ b_0.
 \end{array}$$

Se obține astfel un circuit combinațional care calculează suma $t_{j-1} + a_j + b_j$, denumit **sumator elementar**.

Sumatorul elementar poate fi realizat prin conectarea în cascadă a două semisumatoare SS_1 și SS_2 (fig. 5.11).

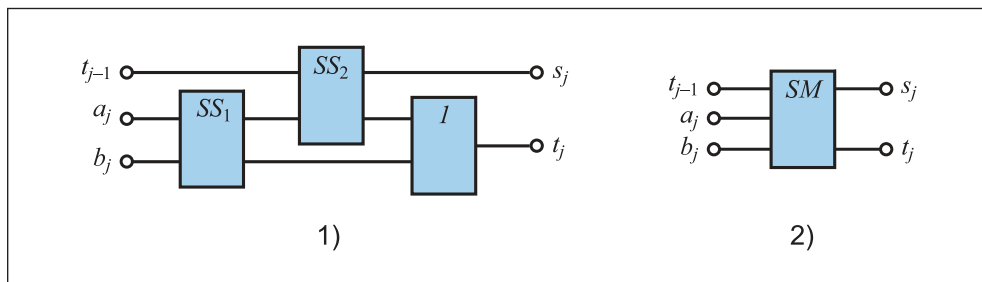


Fig. 5.11. Schema sumatorului elementar (1) și simbolul utilizat (2)

Semisumatorul SS_1 calculează suma $(a_j + b_j)$, iar semisumatorul SS_2 însumează transportul t_{j-1} cu suma $(a_j + b_j)$ calculată de primul semisumator. Transportul t_j către rangul superior $j + 1$ se calculează de poarta logică SAU , care reunește transporturile intermediare de la ieșirile respective ale semisumatoarelor SS_1 și SS_2 .

Suma numerelor binare A și B se calculează cu ajutorul unui circuit combinațional denumit **sumator**. Un sumator poate fi realizat prin conectarea a n sumatoare elementare (fig. 5.12).

Sumatorul elementar SM_0 corespunzător cifrei celei mai puțin semnificative poate fi înlocuit cu un semisumator, deoarece pentru această poziție nu există un transport de la rangul precedent. Transportul de la ieșirea sumatorului SM_{n-1} al rangului cel mai semnificativ este folosit pentru a indica **depășirea capacității** sumatorului de n biți.

Din analiza figurilor 5.10, 5.11 și 5.12 rezultă că un dispozitiv complex – sumatorul de n biți este realizat prin reunirea a unor dispozitive mult mai simple, adică a n sumatoare elementare. Fiecare sumator elementar, la rândul lui, este realizat prin reunirea a câte două semisumatoare și o poartă logică SAU .

Metoda de elaborare a dispozitivelor complexe (de exemplu, sumatorul) prin reunirea mai multor dispozitive identice mai simple (sumatorul elementar) poartă denumirea de **metodă de proiectare ierarhică**. Conform acestei metode, componentele calculatorului se caracterizează prin niveluri de ierarhie, de exemplu:

nivelul 1 – tranzistoare;

nivelul 2 – porți logice;

nivelul 3 – semisumatoare, sumatoare elementare etc.;

nivelul 4 – sumatoare, scăzătoare etc.;

nivelul 5 – unități aritmetice, unități de comandă etc.

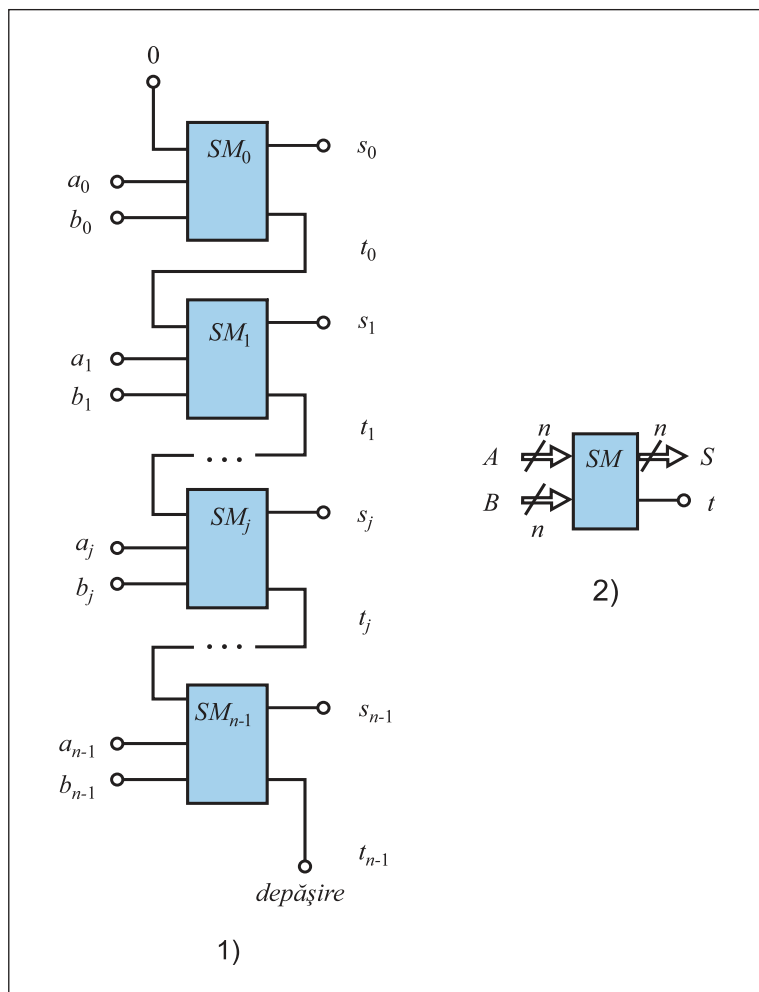


Fig. 5.12. Schema sumatorului (1) și simbolul utilizat (2)

Componentele de un nivel ierarhic inferior sînt utilizate în calitate de „cubuşoare” elementare pentru realizarea componentelor de un nivel ierarhic superior.

Teoretic, dispozitivele unui calculator numeric pot fi elaborate și fără aplicarea metodei de proiectare ierarhică. De exemplu, în cazul sumatorului elementar utilizarea semisumatorului nu este obligatorie. E suficient să alcătuim tabelul de adevăr al sumatorului elementar, să exprimăm funcțiile de ieșire prin formule și să reunim porțile logice respective.

În cazul nivelurilor ierarhice superioare, neaplicarea metodei de proiectare ierarhică face imposibilă elaborarea dispozitivelor complexe. De exemplu, în cazul unui sumator de n biți, tabelul respectiv de adevăr ar conține 2^{2n} rînduri. Pentru $n = 16$ obținem $2^{2 \cdot 16} = 2^{32} \approx 10^9$ rînduri. Evident, formulele funcțiilor de ieșire ale sumatorului de 16 biți practic nu mai pot fi scrise. Prin urmare, sîntem nevoiți să aplicăm metoda de proiectare ierarhică și să realizăm sumatorul printr-o reunire de n sumatoare elementare.

Aplicînd metoda de proiectare ierarhică, într-un mod similar pot fi elaborate circuitele combinaționale destinate scăderii numerelor binare: **semiscăzătorul**, **scăzătorul elementar** și **scăzătorul**.

Întrebări și exerciții

- ❶ Care este destinația semisumatorului? Dar a sumatorului elementar? Și în fine a sumatorului de n biți?
- ❷ Alcătuiți tabelul de adevăr al sumatorului elementar. Tabelul va conține cinci coloane: trei pentru intrările a_j , b_j , t_{j-1} și două pentru ieșirile s_j , t_j .
- ❸ Elaborați un program PASCAL care simulează funcționarea sumatorului elementar. Cifrele binare a_j , b_j și cifra de transport t_{j-1} de la rangul inferior se citesc de la tastatură, iar cifra-sumă s_j și cifra de transport t_j către rangul superior se afișează pe ecran.
- ❹ Explicați esența metodei de proiectare ierarhică a dispozitivelor unui calculator numeric. Este oare obligatorie aplicarea acestei metode? Argumentați răspunsul dvs.
- ❺ Cîte porți logice *NU*, *ȘI*, *SAU* va conține un sumator de 16 biți? Dar de 32 de biți?
- ❻ **Semiscăzătorul** este un circuit combinațional destinat scăderii a două cifre binare. Circuitul respectiv are intrările a , b și ieșirile d , i . Prin termenul d se înțelege diferența $a-b$, iar prin termenul i – împrumutul de la cifra de rang imediat superior. Alcătuiți tabelul de adevăr și elaborați schema semiscăzătorului.
- ❼ **Scăzătorul elementar** este un circuit combinațional capabil de a calcula diferența d_j și împrumutul i_j către rangul imediat superior, dacă se introduce la intrare descăzutul a_j , scăzătorul b_j și împrumutul i_{j-1} de la rangul anterior. Aplicînd metoda de proiectare ierarhică, elaborați schema scăzătorului elementar.
- ❽ Aplicînd metoda de proiectare ierarhică, elaborați schema unui **scăzător** de n biți.
- ❾ Cîte porți logice *NU*, *ȘI*, *SAU* va conține un scăzător de 16 biți? Dar de 32 de biți?
- ❿ Este oare obligatorie aplicarea metodei de proiectare ierarhică în cazul elaborării unui scăzător de n biți? Argumentați răspunsul dvs.

5.4. Circuite combinaționale frecvent utilizate

Circuitele combinaționale frecvent utilizate sînt prezentate în *figura 5.13*.

Sumatorul este un circuit combinațional destinat adunării a două numere binare. Tabelul de adevăr și schema sumatorului au fost studiate în paragraful precedent.

Comparatorul este un circuit combinațional care compară numerele binare A și B , indicînd prin cele trei ieșiri una dintre situațiile posibile: $A < B$, $A > B$ sau $A = B$.

Codificatorul este un circuit combinațional care efectuează conversiunea mesajelor $s_1, s_2, \dots, s_n$ în cuvintele binare din codul respectiv. Se consideră că fiecare dintre mesajele s_i este reprezentat prin valorile $x_1 = 0, \dots, x_i = 1, \dots, x_n = 0$ aplicate la intrarea codificatorului, iar cuvîntul de cod – prin variabilele de ieșire $y_1, y_2, \dots, y_m$.

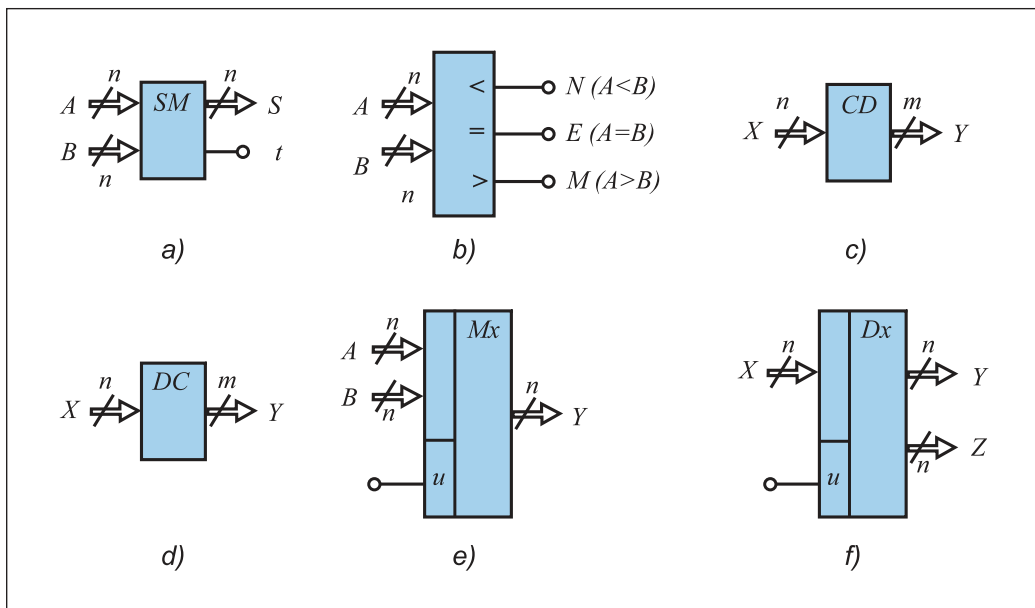


Fig. 5.13. Circuite combinaționale frecvent utilizate:
a – sumatorul; b – comparatorul; c – codificatorul; d – decodificatorul;
e – multiplexorul; f – demultiplexorul

De exemplu, variabilele $x_1, x_2, x_3, \dots$ pot reprezenta starea tastelor $\langle A \rangle, \langle B \rangle, \langle C \rangle, \dots$ ale tastaturii. Codificatorul respectiv va furniza la ieșire cuvîntul de cod *ASCII* corespunzător tastei acționată.

Decodificatorul este un circuit combinațional care generează semnalul logic 1 pe o ieșire distinctă pentru fiecare combinație a valorilor variabilelor de intrare. Cu alte cuvinte, decodificatorul efectuează operația inversă a unui codificator.

Decodificatoarele sînt utilizate pentru a determina operațiile pe care trebuie să le execute procesorul, pentru selectarea unităților de intrare-ieșire, sintetizarea simbolurilor etc.

Multiplexorul este un circuit combinațional destinat selectării fluxurilor de date. În figura 5.13e este prezentat un multiplexor care transmite la ieșire biții numărului binar A ($u = 0$) sau B ($u = 1$). În calculatoarele moderne multiplexoarele se utilizează pentru transferul informației de la mai multe surse la un singur destinatar.

Demultiplexorul distribuie fluxul de date de la intrarea X la una dintre ieșirile Y ($u = 0$) sau Z ($u = 1$). Drept exemplu, amintim transferul informației de la o singură sursă la mai mulți destinatari.

Întrebări și exerciții

- 1 Explicați destinația circuitelor combinaționale frecvent utilizate: sumatorul, comparatorul, codificatorul, decodificatorul, multiplexorul și demultiplexorul.
- 2 Alcătuiți tabelul de adevăr al unui comparator de 2 biți.

- ③ Cîte intrări și cîte ieșiri poate avea un codificator? Cîte intrări și cîte ieșiri poate avea un decodificator?
- ④ Pe panoul de comandă al imprimantei sînt montate butoanele *ON LINE* (funcționare sub controlul unității centrale), *OFF LINE* (funcționare autonomă), *LINE FEED* (avans de linie) și *FORM FEED* (avans de pagină). Alcătuiți tabelul de adevăr al codicatorului care furnizează la ieșire următoarele combinații binare:

00 – *ON LINE*;

01 – *OFF LINE*;

10 – *LINE FEED*;

11 – *FORM FEED*.

- ⑤ Pe panoul de comandă al imprimantei sînt montate becurile (diodele luminescente) indicatoare *READY* (disponibilă), *PAPER* (lipsa de hîrtie), *TEST* (regimul de testare) și *LOAD* (regimul de încărcare a informației). Alcătuiți tabelul de adevăr al decodicatorului care aprinde becurile în cauză. Stările respective sînt codificate prin următoarele combinații binare:

00 – *READY*;

01 – *PAPER*;

10 – *TEST*;

11 – *LOAD*.

- ⑥ Tastatura calculatorului include tastele funcționale $\langle F1 \rangle$, $\langle F2 \rangle$, ..., $\langle F12 \rangle$. Alcătuiți tabelul de adevăr al codicatorului care va furniza la ieșire numărul binar corespunzător tastei funcționale acționate.
- ⑦ Unitățile de intrare-ieșire ale unui calculator au următoarele adrese:

0000 – tastatura;

0001 – vizualizatorul;

0010 – imprimanta mecanică;

0011 – imprimanta cu jet de cerneală;

0100 – imprimanta laser;

0101 – unitatea de disc flexibil *A*;

0110 – unitatea de disc flexibil *B*;

0111 – unitatea de disc rigid *C*;

1000 – unitatea de disc rigid *D*.

Alcătuiți tabelul de adevăr al decodicatorului care va selecta unitatea indicată de adresa respectivă.

- ⑧ Operațiile aritmetice ale unui calculator ipotetic sînt codificate după cum urmează:

<i>adunarea</i>	– 000;
<i>scăderea</i>	– 001;
<i>înmulțirea</i>	– 010;
<i>împărțirea</i>	– 011;
<i>compararea</i>	– 100.

Alcătuieți tabelul de adevăr al decodicatorului operațiilor aritmetice.

- 9 Alcătuieți tabelul de adevăr al multiplexorului cu 2 linii de intrare.

5.5. Bistabilul RS

E cunoscut faptul că într-un circuit secvențial valorile variabilelor de ieșire depind nu numai de combinațiile valorilor variabilelor de intrare, dar și de **consecutivitatea aplicării lor**. Cu alte cuvinte, circuitul secvențial memorizează informații referitoare la combinațiile aplicate la intrări în momentele precedente. Acest lucru este posibil datorită faptului că circuitele secvențiale sînt alcătuite din circuite combinaționale și elemente de **memorie binară**.

Elementul de memorie binară este un circuit cu două stări distincte, destinat pentru a păstra o informație dintr-un singur bit. Circuitul respectiv are denumirea de bistabil.

În figura 5.14 este prezentată schema circuitului bistabil de bază realizat cu porți logice SAU-NU, denumit **bistabil asincron RS**.

Circuitul are două intrări notate cu R și S și două ieșiri notate cu Q și $\bar{Q}$. Se observă că semnalele de ieșire Q și $\bar{Q}$ sînt aplicate la intrările porților SAU-NU. Conexiunile respective sînt numite **reacții**. Tocmai datorită acestor conexiuni, circuitul dat are două stări distincte și, prin urmare, asigură memorarea unui bit de informație.

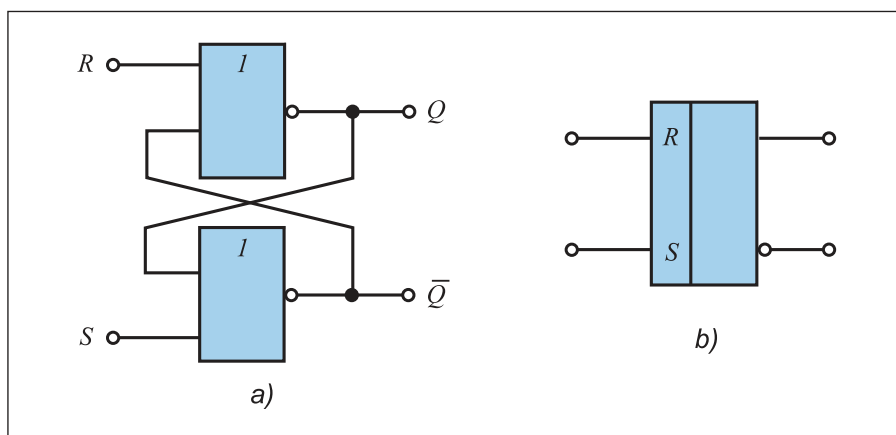


Fig. 5.14. Schema bistabilului asincron RS (a) și simbolul utilizat (b)

Într-adevăr, admitem că semnalele de intrare $R = S = 0$, iar ieșirile $Q = 1$, $\bar{Q} = 0$. Datorită reacției, valoarea $Q = 1$ impune ieșirea $\bar{Q} = 0$. La rândul ei, tot datorită reacției, valoarea $\bar{Q} = 0$ confirmă ieșirea $Q = 1$. Exact la fel, în cazul în care ieșirile $Q = 0$, $\bar{Q} = 1$, datorită reacției, valoarea $\bar{Q} = 1$ impune ieșirea $Q = 0$. Prin urmare, valorile de intrare $R = S = 0$ nu schimbă starea bistabilului, acesta păstrând cifra binară memorată.

S-a convenit ca stării $Q = 1$, $\bar{Q} = 0$ să-i corespundă cifra binară 1, iar stării $Q = 0$, $\bar{Q} = 1$ cifra binară 0. Ieșirea Q este numită ieșire directă sau ieșire adevărată, iar ieșirea $\bar{Q}$ ieșire inversă sau ieșire negată. Starea bistabilului este indicată de ieșirea directă Q .

Să examinăm cazul în care $R = 0$, $S = 1$. Presupunem că bistabilul se află în starea 0, adică $Q = 0$ și $\bar{Q} = 1$. În acest caz, valoarea $S = 1$ va impune $\bar{Q} = 0$, care, la rândul ei, prin reacție, va asigura $Q = 1$. Prin urmare, bistabilul trece în starea 1 ($Q = 1$, $\bar{Q} = 0$). Această stare, după cum s-a stabilit anterior, se va păstra și după trecerea semnalului S de la valoarea 1 la valoarea 0. Intrarea S , care asigură fixarea bistabilului în starea 1, poartă denumirea de **intrare de setare**.

În mod similar se poate constata că în cazul în care bistabilul se află în starea 1 ($Q = 1$, $\bar{Q} = 0$), iar $S = 0$, și $R = 1$, bistabilul va trece în starea 0 ($Q = 0$, $\bar{Q} = 1$). Această stare se va păstra și după trecerea semnalului R de la valoarea 1 la valoarea 0. Intrarea R , care asigură stabilirea bistabilului în starea 0, poartă denumirea de **intrare de resetare**.

Combinăția de intrare $R = 1$ și $S = 1$ impune valorile de ieșire $Q = 0$ și $\bar{Q} = 0$. Valorilor de ieșire $Q = \bar{Q} = 0$ nu le corespunde nicio stare distinctă a bistabilului. Prin urmare, pentru bistabilul RS combinația de intrare $R = S = 1$ este o **combinație interzisă**.

Regimurile de funcționare a bistabilului în cauză sînt totalizate în *tabelul 5.1*.

Tabelul 5.1

Regimurile de funcționare ale bistabilului asincron RS

<i>Intrări</i> R S		<i>Regim de funcționare</i>	<i>Ieșirea Q</i>
0	0	păstrare bit	bit memorat
0	1	setare	1
1	0	resetare	0
1	1	interzis	—

Adjectivul *asincron* din denumirea bistabilului RS concretizează modul de influență a semnalelor de comandă R și S asupra stării bistabilului. Din analiza circuitului prezentat în *figura 5.14* rezultă că semnalele de comandă, aplicate la intrările R și S în momente arbitrare, pot schimba starea bistabilului.

Circuitele secvențiale în care starea circuitului poate fi schimbată de semnalele de comandă în momente arbitrare de timp se numesc circuite asincrone.

Un calculator modern conține zeci de mii de bistabile. Schimbarea stării lor în momente arbitrare, greu de controlat, poate fi cauza unor erori de funcționare. Pen-

tru a evita o funcționare greșită, s-a convenit ca această comportare a circuitelor secvențiale să fie determinată în baza valorilor semnalelor de comandă aplicate la intrări în momente discrete, bine determinate în timp. Aceste momente sînt indicate cu ajutorul unor impulsuri speciale, numite **semnale de sincronizare**.

Circuitele secvențiale în care starea circuitului poate fi schimbată de semnalele de comandă numai în momentele indicate de semnalele de sincronizare se numesc circuite sincrone.

De obicei, semnalul de sincronizare se notează prin litera C (din engleză *Clock* „ceas”) și este furnizat de un dispozitiv special, numit **ceas de sistem**.

În figura 5.15 este prezentată schema unui bistabil sincron denumit **bistabil sincron RS**.

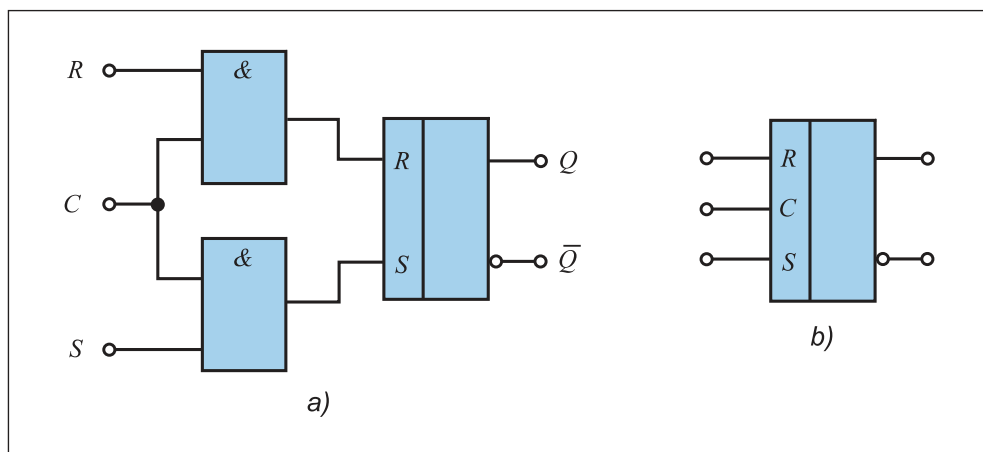


Fig. 5.15. Schema bistabilului sincron RS (a) și simbolul utilizat (b)

Schema dată include bistabilul asincron RS (fig. 5.14) și două porți logice ȘI care permit aplicarea semnalelor de comandă la intrările bistabilului asincron numai în cazul în care semnalul de sincronizare C are valoarea 1.

Întrebări și exerciții

- ❶ Prin ce se deosebesc circuitele combinaționale și circuitele secvențiale?
- ❷ Care este destinația circuitului bistabil?
- ❸ Cum funcționează un circuit bistabil cu porți logice SAU-NU? Care este destinația reacțiilor?
- ❹ Explicați regimurile de funcționare a bistabilului asincron RS. De ce combinația $R = S = 1$ nu poate fi aplicată la intrările bistabilului examinat?
- ❺ Prin ce se deosebesc circuitele secvențiale asincrone și circuitele secvențiale sincrone?
- ❻ Explicați cum funcționează bistabilul sincron RS. Care este destinația porților logice ȘI din componența acestui bistabil?

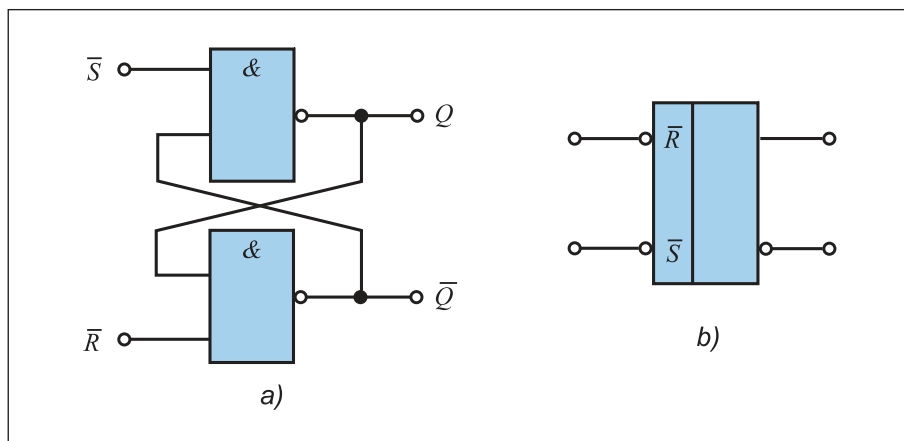


Fig. 5.16. Schema bistabilului asincron $\bar{R}\bar{S}$ (a) și simbolul utilizat (b)

- 7 În figura 5.16 este prezentată schema circuitului bistabil de bază realizat cu porți logice ȘI-NU, denumit **bistabil asincron $\bar{R}\bar{S}$** .

Circuitul are următoarele regimuri de funcționare:

- păstrare bit ($\bar{R} = 1, \bar{S} = 1$);
- setare ($\bar{R} = 1, \bar{S} = 0$);
- resetare ($\bar{R} = 0, \bar{S} = 1$).

Explicați cum funcționează bistabilul examinat. De ce combinația de intrare $\bar{R} = 0, \bar{S} = 0$ este o combinație interzisă?

- 8 Utilizând bistabilul asincron $\bar{R}\bar{S}$, elaborați schema **bistabilului sincron $\bar{R}\bar{S}$** .
- 9 Desenați schema detaliată (la nivelul porților logice ȘI, SAU-NU) a bistabilului sincron $\bar{R}\bar{S}$. Pentru aceasta decalcați schemele respective din figurile 5.14a și 5.15a.
- 10 Desenați schema detaliată (la nivelul porților logice ȘI, ȘI-NU) a bistabilului sincron $\bar{R}\bar{S}$.

5.6. Circuite secvențiale frecvent utilizate

Registrul (fig. 5.17a) este un dispozitiv numeric destinat păstrării temporare a unui număr binar,

$$D = d_{n-1} \dots d_1 d_0.$$

Registrul este constituit din bistabile la care sînt atașate circuite combinaționale care permit înscrierea, citirea sau transferul informației. Înscrierea informației în registru se efectuează prin aplicarea la intrarea W (Write „înscrie”) a impulsului respectiv. Deoarece fiecare bistabil poate memora un singur bit, **capacitatea n** a unui registru este dată de numărul bistabilelor.

În anumite aplicații, de exemplu: înmulțirea și împărțirea numerelor binare, scrierea sau citirea datelor pe disc, transmiterea datelor prin firele telefonice etc., apare necesitatea deplasării în stînga sau în dreapta a informației memorate într-un registru. În acest scop se utilizează **registrele de deplasare** (fig. 5.17 b, c).

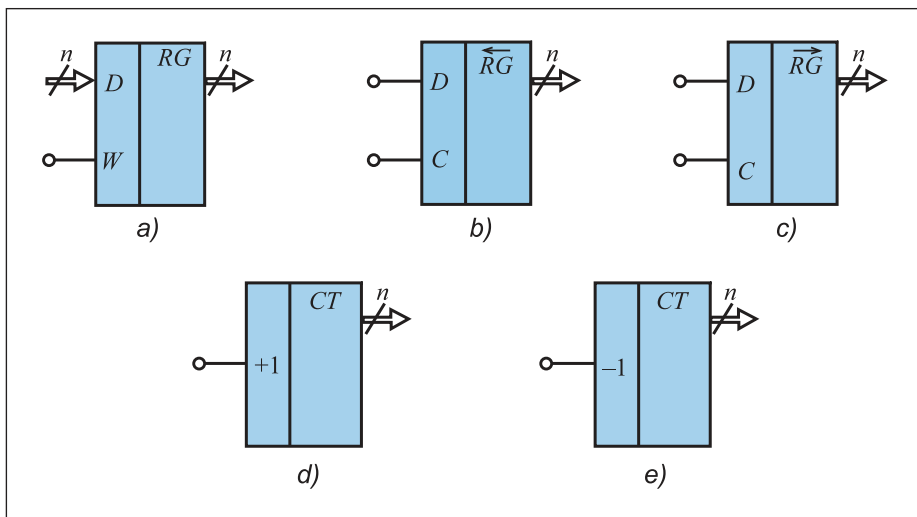


Fig. 5.17. Circuite secvențiale frecvent utilizate:

a – registru; b – registru de deplasare de la dreapta spre stînga; c – registru de deplasare de la stînga spre dreapta; d – numărător direct; e – numărător invers

Succesiunea stărilor unui **registru de deplasare de la dreapta spre stînga** este dată în figura 5.18.

<i>Timpul</i>	d_3	d_2	d_1	d_0
inițial	0	1	0	1
t_1	1	0	1	0
t_2	0	1	0	0
t_3	1	0	0	0
t_4	0	0	0	0
t_5	0	0	0	0
...		...		

Fig. 5.18. Succesiunea stărilor unui registru de deplasare de la dreapta spre stînga

Se consideră că starea inițială a registrului este 0101, iar impulsurile sînt aplicate la intrarea C în momentele de timp t_1 , t_2 , t_3 etc. Firește, registru în cauză calculează produsul $D \times 2$.

În mod similar, un **registru de deplasare de la stînga spre dreapta** va calcula cîtul $D : 2$.

Numărătoarele sînt circuite secvențiale care înregistrează numărul de impulsuri aplicat la intrare. Numărătoarele se clasifică după următoarele criterii:

- modul de codificare a informației (binar, binar-zecimal etc.);
- modul de modificare a stărilor numărătorului (numărătoare directe, inverse și reversibile).

În general, numărătoarele sînt realizate prin asocierea circuitelor bistabile cu circuite combinaționale care determină modul corect în care numărătoarele urmează să-și modifice starea la fiecare nou impuls sosit la intrare.

Un **numărător binar** înregistrează succesiunea impulsurilor aplicate la intrare în sistemul de numerație binar. **Capacitatea de numărare** a numărătorului binar depinde de numărul bistabilelor. Considerînd pentru fiecare număr o stare distinctă, rezultă că acesta poate număra în gama 0 la $2^n - 1$, n fiind numărul bistabilelor.

În figura 5.17d este prezentat **numărătorul binar direct**, iar în figura 5.19 tabelul succesiunii stărilor unui numărător binar direct de 3 biți.

<i>Timpul</i>	d_2	d_1	d_0
inițial	0	0	0
t_1	0	0	1
t_2	0	1	0
t_3	0	1	1
t_4	1	0	0
t_5	1	0	1
t_6	1	1	0
t_7	1	1	1
t_8	0	0	0
t_9	0	0	1
...	...		

Fig. 5.19. Succesiunea stărilor unui numărător binar direct de trei biți

Numărătoarele care își schimbă starea conform tabelului din figura 5.19 sînt denumite **directe**, deoarece conținutul numărătorului crește cu o unitate, la fiecare nou impuls sosit la intrarea +1. Dacă într-un numărător se introduce inițial un număr și la fiecare impuls aplicat la intrarea -1 conținutul său scade cu o unitate, se obține un **numărător invers** (fig. 5.17e).

Întrebări și exerciții

- 1 Care este destinația registrului? De ce depinde capacitatea unui registru?
- 2 În registrul de deplasare de la dreapta spre stînga (fig. 5.17b) este încărcat numărul binar 1001. Care va fi conținutul registrului după aplicarea la intrarea C a unui impuls? Dar a două impulsuri?
- 3 În registrul de deplasare de la stînga spre dreapta este încărcat unul dintre următoarele cuvinte binare:

a) 00000;

f) 00001;

b) 10000;

g) 10001;

c) 01000;

h) 01010;

d) 00100;

i) 01100;

e) 00010;

j) 00110.

Care va fi conținutul registrului după aplicarea la intrarea C a două impulsuri consecutive?

- ④ Elaborați un program PASCAL care simulează funcționarea unui registru de deplasare de la stînga spre dreapta de n biți.
- ⑤ Care este destinația numărătoarelor? Cum se schimbă stările unui numărător direct? Dar ale unui numărător invers?
- ⑥ Un numărător direct de 4 biți se află inițial în una dintre următoarele stări:

- | | |
|----------|----------|
| a) 0000; | f) 1010; |
| b) 0010; | g) 1100; |
| c) 0100; | h) 1111; |
| d) 1000; | i) 0101; |
| e) 1001; | j) 0110. |

Care va fi starea numărătorului după aplicarea a 5 impulsuri de intrare? Dar a 8 impulsuri?

- ⑦ Un numărător invers de 4 biți se află în starea inițială 1001. Care va fi starea numărătorului după aplicarea a m impulsuri de intrare? Numărul m poate avea valorile 1, 4, 5, 8, 11, 17.
- ⑧ Elaborați un program PASCAL care simulează funcționarea unui numărător direct de n biți.

5.7. Generatoare de impulsuri

Impulsurile se utilizează în echipamentele numerice pentru a asigura funcționarea secvențială a acestora. De obicei, generatoarele de impulsuri se realizează pe baza porților logice și a elementelor de întârziere.

Elementul de întârziere reprezintă un circuit electronic care realizează funcția logică de repetare $y = x$, însă semnalul de ieșire y repetă semnalul de intrare x cu o întârziere de Δ unități de timp.

Din cursul de fizică e cunoscut faptul că viteza de propagare a semnalelor este finită. Prin urmare, orice conductor poate fi tratat ca un element de întârziere. Pentru a mări „inertitatea” circuitelor electronice și pentru a realiza întârzieri semnificative, în componența lor se includ condensatoare și rezistoare. În acest caz întârzierea Δ este determinată de capacitatea și rezistența componentelor respective.

În *figura 5.20* sînt prezentate simbolul și diagramele în timp ale elementului de întârziere.

Schema unui **generator de impulsuri periodice** realizat pe baza unui element de întârziere și a porții logice *ȘI-NU* este prezentată în *figura 5.21*. În starea inițială $x = 0$ și $y = 1$, iar la ieșirea elementului de întârziere se menține valoarea logică 1. Cînd la intrare se aplică semnalul de pornire $x = 1$, ieșirea devine egală cu 0 (*fig. 5.22*). În continuare, valoarea logică 0, după o întârziere Δ , este aplicată la a doua intrare a porții logice *ȘI-NU*. Astfel, ieșirea y devine egală cu 1. Valoa-

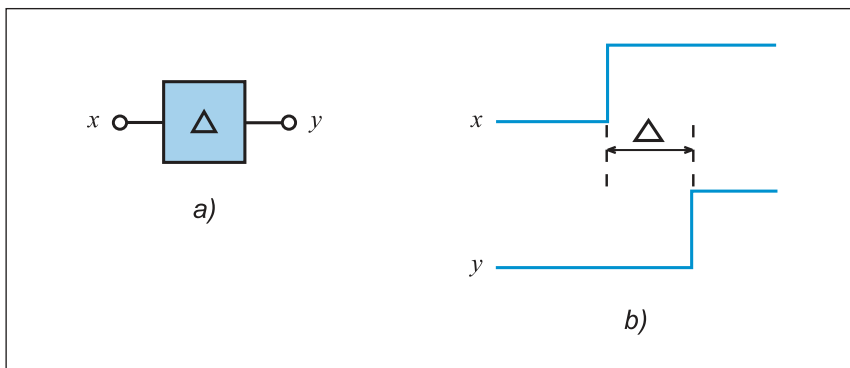


Fig. 5.20. Simbolul (a) și diagramele în timp (b) ale elementului de întârziere

rea logică 1, după o întârziere Δ , va fi din nou aplicată la intrarea porții logice ȘI-NU, impunând la ieșire valoarea $y = 0$ etc.

Prin urmare, la ieșirea y a generatorului se va forma o succesiune de impulsuri cu durata Δ . Procesul de generare poate fi întrerupt prin aplicarea la intrare a semnalului de oprire $x = 0$.

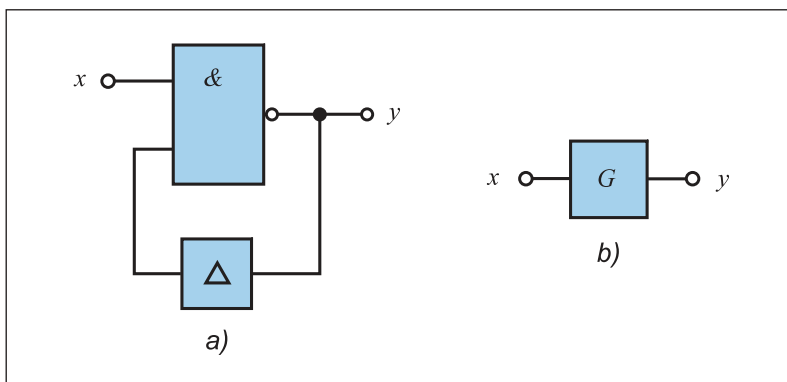


Fig. 5.21. Schema (a) și simbolul generatorului de impulsuri periodice (b)

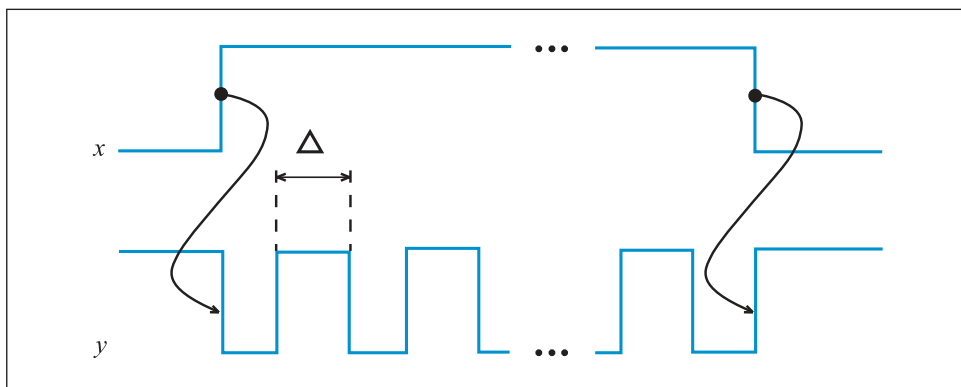


Fig. 5.22. Diagramele în timp ale generatorului de impulsuri periodice

Întrebări și exerciții

- 1 Care este destinația elementului de întârziere? Desenați diagramele în timp ale elementului examinat.
- 2 Explicați cum funcționează generatorul de impulsuri periodice. De ce depinde durata impulsurilor?
- 3 Înlocuiți poarta logică *SI-NU* din componența generatorului de impulsuri periodice prezentat în *figura 5.21* printr-o poartă logică *SAU-NU*. Explicați cum va funcționa circuitul respectiv. Desenați diagramele în timp ale generatorului obținut.
- 4 E cunoscut faptul că variațiile mărimilor fizice nu pot avea loc instantaneu. Prin urmare, orice poartă logică are o întârziere δ , denumită **întârziere parazită**, valoarea concretă a căreia depinde de particularitățile circuitului respectiv.

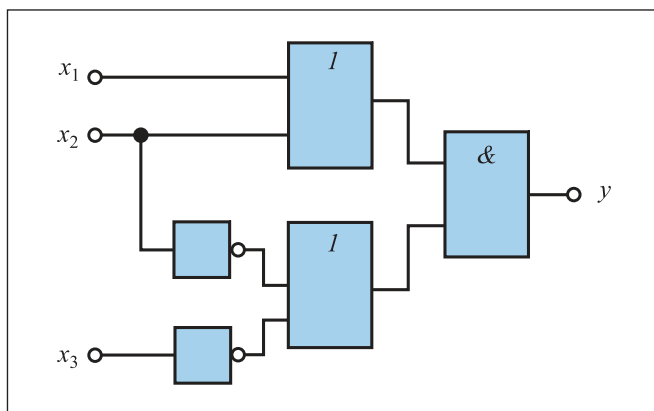
Exclueți din schema prezentată în *figura 5.21* elementul de întârziere, aplicând semnalul de la ieșirea porții logice *SI-NU* direct la intrarea ei. Explicați cum va funcționa circuitul obținut. De ce depinde durata impulsurilor la ieșirea porții logice? Desenați diagramele respective în timp.

- 5 Circuitul secvențial constă dintr-o poartă logică *NU*, semnalul de ieșire al căreia este aplicat direct la intrarea ei. Cum va funcționa acest circuit?

Test de autoevaluare nr. 5

1. Desenați schema circuitului logic pentru calcularea funcției $y = \bar{x}_1 x_2 \vee x_1 x_3$.

2. La intrările circuitului logic de mai jos sînt aplicate valorile $x_1 = 1$, $x_2 = 0$ și $x_3 = 1$. Desenați schema în caiet și indicați pe desen valorile semnalelor de la intrările și ieșirile fiecărei porți logice.



3. Scrieți funcția logică realizată de circuitul din itemul 2.
4. Elaborați un program PASCAL care simulează funcționarea semisumatorului. Cifrele binare a și b se citesc de la tastatură, iar cifra-sumă s și cifra de transport t se afișează pe ecran.

5. Cîte porți logice *NU*, *ȘI*, *SAU* va conține un sumator de 8 biți? Argumentați răspunsul.

6. Elaborați un program PASCAL care simulează funcționarea sumatorului de 8 biți. Numerele binare *A* și *B* se citesc de la tastatură, iar suma *S* și cifra de depășire *t* se afișează pe ecran.

7. Indicați corespondența între denumirile circuitelor combinaționale frecvent utilizate (coloana din stînga) și destinația acestora (coloana din dreapta):

- | | |
|----------------------|--|
| | (a) efectuează conversiunea mesajelor în cuvinte binare; |
| (1) sumatorul; | (b) efectuează conversiunea semnalelor; |
| (2) comparatorul; | (c) distribuie fluxurile de date; |
| (3) codificatorul; | (d) efectuează conversiunea cuvintelor binare în mesaje; |
| (4) decodificatorul; | (e) calculează suma a două numere binare; |
| (5) multiplexorul; | (f) selectează obiectele unei imagini; |
| (6) demultiplexorul. | (g) compară două numere binare; |
| | (h) selectează fluxurile de date. |

8. Pe panoul de comandă al unui joc electronic sînt montate becurile (diodele luminiscente) indicatoare SUS, JOS, STÎNGA și DREAPTA. Alcătuiți tabelul de adevăr al decodificatorului care aprinde becurile în cauză. Comenzile respective sînt codificate prin următoarele combinații binare:

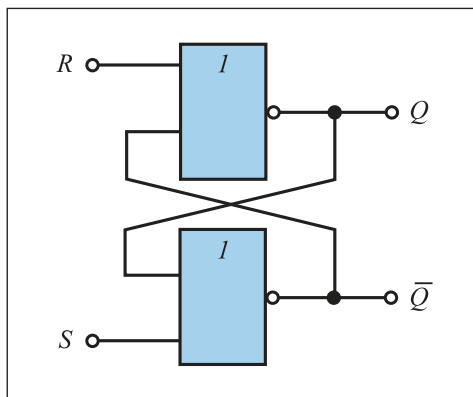
00 – SUS;

01 – JOS;

10 – STÎNGA;

11 – DREAPTA.

9. Cunoscînd valorile variabilelor de intrare $R = 0$ și $S = 1$, determinați valorile funcțiilor Q și $\bar{Q}$ la ieșirea bistabilului de mai jos.



10. Desenați schema detaliată (la nivelul porților logice *ȘI*, *ȘI-NU*) a bistabilului sincron $\bar{R}\bar{S}$.

11. Indicați corespondența între denumirile circuitelor secvențiale frecvent utilizate (coloana din stînga) și destinația acestora (coloana din dreapta):

	(a)	deplasează cuvîntul binar de la stînga la dreapta;
(1) bistabilul;	(b)	transformă cuvintele binare în caractere;
(2) registrul;	(c)	adună cîte o unitate la conținutul său;
(3) registrul de deplasare de la dreapta spre stînga;	(d)	memorează o cifră binară;
(4) registrul de deplasare de la stînga spre dreapta;	(e)	adună două numere binare, reprezentate în cod invers;
(5) numărătorul direct;	(f)	scade cîte o unitate din conținutul său;
(6) numărătorul invers.	(g)	deplasează cuvîntul binar de la dreapta la stînga;
	(h)	scade numerele binare, reprezentate în cod direct;
	(i)	memorează un cuvînt binar.

12. În registrul de deplasare de la stînga spre dreapta este încărcat cuvîntul binar 10011011. Care va fi conținutul registrului după aplicarea la intrarea *C* a trei impulsuri consecutive?

13. În registrul de deplasare de la dreapta spre stînga este încărcat cuvîntul binar 11011101. Care va fi conținutul registrului după aplicarea la intrarea *C* a patru impulsuri consecutive?

14. Elaborați un program PASCAL care simulează funcționarea unui registru de deplasare de la dreapta spre stînga. Conținutul inițial al registrului *R* și numărul de impulsuri de intrare *m* se citesc de la tastatură, iar rezultatul deplasării se afișează pe ecran.

15. Un numărător direct de 8 biți se află în starea inițială 10001101. Care va fi starea numărătorului după aplicarea a 6 impulsuri de intrare?

16. Un numărător invers de 8 biți se află în starea inițială 10101110. Care va fi starea numărătorului după aplicarea a 5 impulsuri de intrare?

17. Elaborați un program PASCAL care simulează funcționarea unui numărător direct. Conținutul inițial al numărătorului *A* și numărul de impulsuri de intrare *m* se citesc de la tastatură, iar conținutul final al numărătorului se afișează pe ecran.

Capitolul 6

STRUCTURA ȘI FUNCȚIONAREA CALCULATORULUI

6.1. Schema funcțională a calculatorului

Schema funcțională a calculatorului numeric este prezentată în *figura 6.1*.

Conform acestei scheme, calculatorul numeric conține următoarele **unități funcționale**:

- o unitate de memorie pentru a înmagazina datele inițiale, intermediare și finale ale problemei, precum și instrucțiunile care indică secvența calculelor;
- un dispozitiv aritmetic și logic necesar efectuării operațiilor aritmetice și logice elementare;
- unul sau mai multe dispozitive de intrare, respectiv, ieșire, necesare comunicării din exterior cu calculatorul;
- un dispozitiv central de comandă și control care generează o succesiune de semnale de comandă necesare executării secvențiale a instrucțiunilor.

Dispozitivul aritmetic și logic și dispozitivul central de comandă formează **unitatea centrală de prelucrare a informației** sau, mai pe scurt, **procesorul**.

Memoria calculatoarelor moderne este organizată în două niveluri, și anume: o unitate de memorie internă cu o viteză mare de lucru și una sau mai multe unități de memorie externă cu o viteză mai redusă, însă cu o capacitate mult mai mare decât cea a memoriei interne.

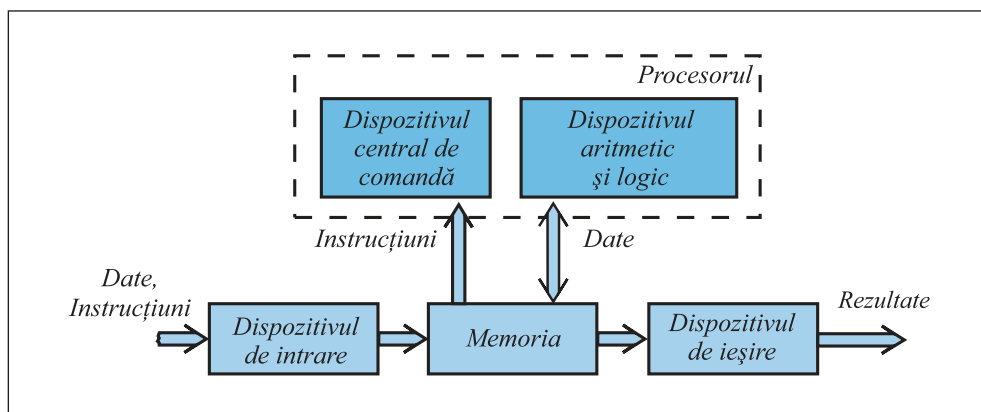


Fig. 6.1. Schema funcțională a calculatorului

Memoria internă (numită uneori memorie principală, centrală sau operativă) păstrează programul în curs de executare și datele folosite de acesta, prezența ei fiind o condiție esențială pentru funcționarea calculatorului.

Memoria externă are rolul de a păstra cantități mari de informație și programe folosite frecvent pentru a putea fi aduse într-un interval de timp mic în memoria internă. În prezent ca memorii externe sînt utilizate unitățile cu discuri sau benzi magnetice, unitățile cu discuri optice etc.

Unitățile de memorie externă și dispozitivele de intrare-ieșire sînt numite **echipamente periferice**.

Pentru a asigura o interacțiune eficientă a procesorului, a memoriei interne și a echipamentelor periferice, în cazul calculatoarelor personale schema funcțională se realizează fizic conform schemei-bloc prezentate în *figura 6.2*.

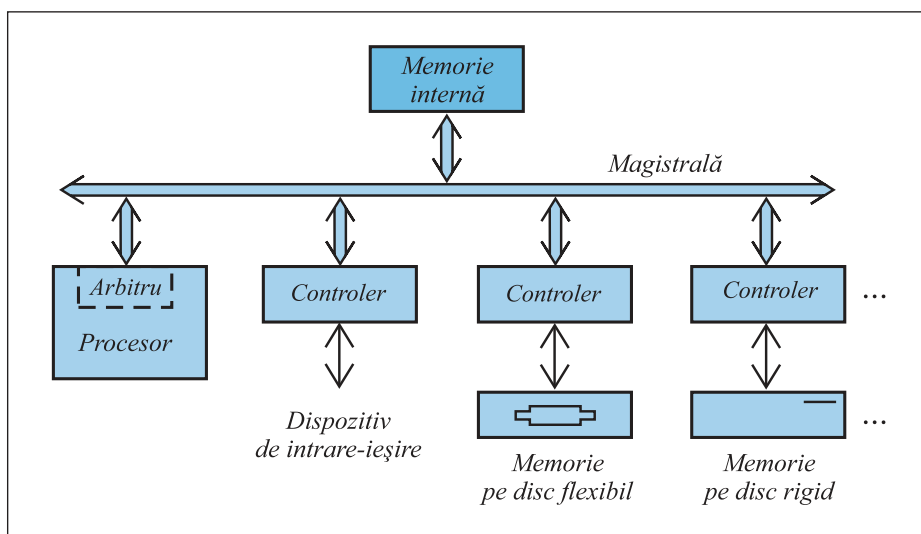


Fig. 6.2. Schema-bloc a unui calculator personal

Din analiza *figurii 6.2* rezultă că toate calculatoarele moderne au o **configurație modulară**. Fiecare modул, și anume controlerul, imprimantele, unitățile de disc magnetic etc., funcționează și, în consecință, pot fi incluse sau excluse din componența calculatorului independent unul de altul. Prin urmare, configurația calculatorului poate fi modificată în funcție de destinația sistemului de calcul.

De exemplu, un sistem editorial va include mai multe tipuri de imprimante: mecanice pentru textele în curs de prelucrare, laser sau color pentru paginile machetate deja, cititoare de desene și fotografii (scanere) etc. Un sistem destinat gestionării rapide a unui volum mare de date va include mai multe discuri magnetice, iar un calculator utilizat pentru montarea filmelor video va fi dotat cu camere de luat vederi și vizualizatoare de o rezoluție adecvată, cu tastaturi similare pupitrului regizoral etc.

Întrebări și exerciții

- ❶ Numiți unitățile funcționale ale calculatorului și explicați destinația lor.
- ❷ Care este rolul memoriei interne? Cum se realizează memoria externă a calculatoarelor moderne?
- ❸ Numiți echipamentele periferice pe care le cunoașteți dvs.
- ❹ Numiți componentele unui calculator personal și explicați destinația lor.
- ❺ Care este structura și cum interacționează componentele unui calculator?
- ❻ Cum poate fi modificată configurația unui sistem de calcul? Care sînt avantajele configurației modulare a calculatorului?
- ❼ Desenați schema-bloc a calculatorului cu care lucrați dvs. Care componente sînt obligatorii și care opționale pentru funcționarea calculatorului?
- ❽ Cum poate fi conectată la un calculator personal o unitate suplimentară de disc magnetic? O imprimantă laser? Un cititor de documente? O cameră de luat vederi?

6.2. Formatul instrucțiunilor

Pentru a rezolva o problemă, calculatorul trebuie să cunoască în fiecare moment atît operația pe care urmează să o execute, cît și datele care participă în operație. Aceste operații sînt comunicate calculatorului prin intermediul instrucțiunilor.

Instrucțiunea calculatorului reprezintă o succesiune de cifre binare prin care se indică procesorului operația de executat și amplasamentul (locul) operanzilor.

Succesiunea binară respectivă, denumită uneori și **cuvînt instrucțiune**, este împărțită în cîmpuri, fiecare cîmp avînd o semnificație precisă. Numărul și semnificația cîmpurilor poartă denumirea de **formatul instrucțiunii**. În *figura 6.3* sînt prezentate formatele utilizate în calculatoarele moderne.

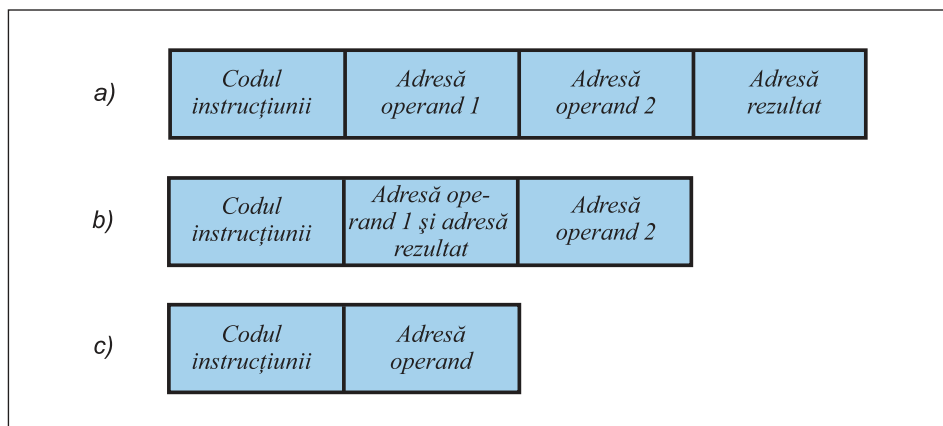


Fig. 6.3. Formatul instrucțiunilor cu trei (a), două (b) și o singură adresă (c)

În general, pentru executarea unei operații este necesar ca instrucțiunea să conțină trei adrese (*fig. 6.3a*). Primele două adrese sînt folosite pentru obținerea celor doi operanzi asupra cărora se va efectua operația specificată de cîmpul *Codul instrucțiunii*. Rezultatul operației va fi depus pe adresa specificată de cîmpul *Adresă rezultat*.

Să analizăm un exemplu. Presupunem că operațiile aritmetice și logice sînt codificate după cum urmează (pentru simplitate vom utiliza echivalentele zecimale ale cîmpurilor binare respective):

01 – adunarea;

02 – scăderea;

03 – operația logică *ȘI*;

04 – operația logică *SAU*.

Instrucțiunea

01 100 110 215

va impune procesorul să adune numerele din locațiile 100 și 110 și să depună suma obținută în locația 215.

Instrucțiunea

02 100 110 215

comunică procesorului că din numărul înscris în locația 100 se scade numărul din locația 110. Rezultatul obținut va fi depus în locația 215.

În mod similar, instrucțiunea

03 200 300 100

specifică operația logică *ȘI* asupra biților cuvintelor din locațiile 200 și 300. Rezultatul operației va fi depus în locația 100.

Se observă că într-o instrucțiune nu se specifică **valoarea operanzilor**, ci **adresele locațiilor** în care pot fi găsiți operanzii respectivi. Acest fapt permite utilizarea unuia și aceluiași program pentru prelucrarea oricăror date inițiale. Faptul că instrucțiunile lucrează cu adrese al căror conținut trebuie prelucrat, și nu cu însuși conținutul, constituie un principiu fundamental al calculatoarelor numerice, care permite ca un program să fie elaborat și introdus în calculator independent de datele concrete asupra cărora se aplică.

În formatul cu trei adrese (*fig. 6.3a*), adresele sînt **specificate explicit**. Pentru o reprezentare mai compactă a instrucțiunilor se utilizează **specificarea implicită** a unor adrese. În acest caz, cuvîntul instrucțiune nu conține un cîmp special pentru adresa implicită.

Dacă rezultatul obținut în urma executării unei instrucțiuni se depune pe adresa unuia dintre operanzi, formatul respectiv va avea numai două adrese (*fig. 6.3b*). Prin urmare, adresa rezultatului este specificată implicit. De exemplu, instrucțiunea

01 100 110

va impune procesorul să adune numerele din locațiile 100 și 110 și să depună suma obținută în locația 100. Evident, după înscrierea sumei, numărul inițial din locația 100 va fi suprimat.

S-a constatat că formatul cu două adrese, în prezent cel mai răspândit, permite scrierea de programe avînd un număr de instrucțiuni comparabil cu cel obținut atunci cînd s-ar utiliza mai multe adrese.

Formatul cu o singură adresă (fig. 6.3c) se utilizează în calculatoarele procesorului cărora include un registru special, denumit **acumulator**. În acumulator se păstrează primul operand și se depune rezultatul executării operației respective. Prin urmare, adresa primului operand și adresa rezultatului sînt specificate implicit. De exemplu, instrucțiunea cu o singură adresă

01 100

va aduna numărul din acumulator cu numărul din locația 100, iar suma obținută va fi depusă în acumulator. Respectiv, numărul inițial din acumulator va fi suprimat.

Instrucțiunile cu o singură adresă sînt eficiente din punctul de vedere al lungimii cuvîntului și al rapidității calculatorului. Totuși un program scris cu instrucțiuni avînd o singură adresă va conține mai multe instrucțiuni decît în cazul în care se folosesc instrucțiuni cu două sau trei adrese.

Menționăm că toate calculatoarele moderne pot avea instrucțiuni de diferite formate. Informația referitoare la formatul fiecărei instrucțiuni se indică în cîmpul *Codul instrucțiunii*.

Întrebări și exerciții

- 1 Enumerați formatele instrucțiunilor utilizate în calculatoarele moderne. Explicați modul de specificare implicită a unor adrese.
- 2 Explicați semnificația cîmpurilor instrucțiunilor cu trei sau două adrese.
- 3 Cum se specifică adresele operandilor și ale rezultatului în cazul instrucțiunilor cu o singură adresă?
- 4 Explicați cum vor fi executate următoarele instrucțiuni cu trei adrese:

a) 01 200 201 202;

c) 03 100 150 250;

b) 04 202 201 200;

d) 02 250 300 310.

Operațiile aritmetice și logice sînt codificate ca și în exemplul din paragraful de față.

- 5 Care va fi conținutul locației 100 după executarea instrucțiunii

01 200 300 100,

dacă în locațiile 200 și 300 sînt înscrise numerele 17 și, respectiv, 31?

- 6 Explicați cum vor fi executate următoarele instrucțiuni cu două adrese:

a) 01 200 201;

c) 03 100 150;

b) 04 202 201;

d) 02 250 300.

- 7 Care va fi conținutul locației 200 după executarea instrucțiunii

01 200 100,

dacă în locațiile 100 și 200 sînt înscrise numerele 18 și, respectiv, 32?

⑧ Explicați cum vor fi executate următoarele instrucțiuni cu o singură adresă:

a) 01 100;

c) 02 400;

b) 03 200;

d) 04 150.

⑨ Care va fi conținutul acumulatorului după executarea instrucțiunii

01 100,

dacă inițial în locația 100 era înscris numărul 12, iar în acumulator numărul 26?

⑩ Care sînt avantajele și dezavantajele formatelor cu trei, două sau cu o singură adresă?

6.3. Tipuri de instrucțiuni

Instrucțiunile unui calculator se împart în patru grupe:

- instrucțiuni operaționale, care efectuează operații aritmetice și logice asupra datelor specificate prin operanzi;
- instrucțiuni de transfer, care deplasează informația între registre și/sau locații fără a modifica informația transferată;
- instrucțiuni de salt, care în urma verificării unor condiții, modifică analiza și execuția secvențială a instrucțiunilor din program;
- instrucțiuni de intrare-ieșire care permit comunicarea calculatorului cu exteriorul.

Instrucțiunile operaționale prelucrează datele păstrate în locațiile memoriei interne și în registrele procesorului. Cele mai cunoscute instrucțiuni ale acestei grupe sînt cele care efectuează operațiile aritmetice de bază: *adunarea*, *scăderea*, *înmulțirea* și *împărțirea*.

Instrucțiunile logice de tipul *ȘI*, *SAU*, *NU* sînt instrucțiuni operaționale care acționează asupra pozițiilor individuale ale informației binare. În categoria instrucțiunilor operaționale întîlnim și instrucțiuni de tipul: *șterge* conținutul unei locații sau al unui registru, *complementează* conținutul unei locații, *crește* cu o unitate conținutul unui registru etc. În fine, în categoria instrucțiunilor operaționale sînt incluse instrucțiunile de *deplasare* a informației, în care partea de adresă a instrucțiunii conține un număr întreg, care specifică numărul pozițiilor cu care se face deplasarea.

Instrucțiunile de transfer deplasează informația dintre locațiile memoriei interne, între registre sau între locații și registre fără a altera conținutul informației transferate. Instrucțiunea trebuie să specifice explicit sau implicit adresa-sursă și adresa de destinație a transferului. În timpul transferului și după transfer, informația din sursă rămîne neschimbată. Cele mai uzuale instrucțiuni ale acestei grupe sînt cele prin care conținutul unei locații trece într-un anumit registru, registrul acumulator, precum și instrucțiunea de transfer invers: dintr-un registru într-o locație a memoriei interne.

Instrucțiunile de salt se utilizează pentru modificarea ordinii de execuție a instrucțiunilor. În mod normal, instrucțiunile unui program sînt analizate și execu-

tate în mod secvențial, în ordinea în care sînt așezate în memorie. Această ordine poate fi schimbată cu ajutorul instrucțiunilor de salt condiționat sau necondiționat.

Instrucțiunile de salt condiționat permit alegerea continuării programului pe o anumită ramură, în funcție de o condiție de test realizată. Folosirea instrucțiunilor de salt condiționat dau posibilitate utilizatorului să introducă decizii logice în procesul execuției programului.

O instrucțiune de salt necondiționat conține, în partea de adresă, adresa instrucțiunii care va fi executată în continuare.

Instrucțiunile de intrare-ieșire permit comunicarea calculatorului cu echipamentele periferice. Echipamentul cu care se va efectua operația de intrare-ieșire se specifică în partea de adresă a instrucțiunii. De regulă, instrucțiunile de acest tip conțin atît informații legate de natura schimbului de date, adică introducerea sau extragerea lor, cît și comenzi necesare funcționării corecte a periferiei. Tot în aceste instrucțiuni se specifică și registrele sau locațiile în care vor fi depuse sau din care vor fi luate datele respective.

Întrebări și exerciții

- ❶ Cum se clasifică instrucțiunile unui calculator? Care este destinația instrucțiunilor din fiecare grupă?
- ❷ Dați cîteva exemple de instrucțiuni operaționale. Estimați numărul instrucțiunilor operaționale posibile.
- ❸ Care este destinația instrucțiunilor de transfer? Estimați numărul instrucțiunilor posibile de transfer.
- ❹ Cînd și cum se utilizează instrucțiunile de salt? Care condiții de test pot fi analizate de aceste instrucțiuni?
- ❺ Care este destinația instrucțiunilor de intrare-ieșire? Ce informații conțin aceste instrucțiuni?

6.4. Limbajul cod calculator și limbajul de asamblare

Pentru a rezolva o problemă, în memoria calculatorului trebuie să fie încărcate programul respectiv și datele de prelucrat. Instrucțiunile programului și datele de prelucrat se înmagazinează în memoria internă sub forma unor succesiuni de cifre binare pe care dispozitivul central de comandă le poate extrage și interpreta.

Programele reprezentate în formă de succesiuni binare direct executabile de calculator se numesc programe în limbaj cod calculator sau programe în limbaj mașină.

Pentru utilizator programul în cod calculator poate fi prezentat în formă de siruri de cifre binare sau, mai compact, de cifre octale, zecimale sau hexazecimale organizate pe locații ale memoriei.

Elaborarea programelor în limbaj cod calculator este un lucru extenuant și inefficient. Pentru a simplifica procesul de elaborare a programelor, s-a convenit ca instrucțiunile să fie scrise într-un limbaj simbolic, denumit **limbaj de asamblare**. În acest limbaj codurile instrucțiunilor se reprezintă printr-un grup de caractere, astfel ales încît să sugereze cît mai bine natura operației. Acest grup de caractere, de regulă trei, este cunoscut sub numele **mnemonica instrucțiunii**.

De exemplu, codurile instrucțiunilor calculatorului ipotetic din paragraful precedent pot fi notate simbolic conform *tabelului 6.1*.

Tabelul 6.1

Mnemonica instrucțiunilor

<i>Cod instrucțiune</i>	<i>Mnemonica</i>	<i>Semnificația</i>
01	INC	<i>Încarcă acumulatorul</i>
02	MEM	<i>Memorează acumulatorul</i>
03	ADU	<i>Adunare</i>
04	SCD	<i>Scădere</i>
05	SLT	<i>Salt necondiționat</i>
06	SLTC	<i>Salt condiționat</i>
07	STP	<i>Stop</i>

Adresele locațiilor memoriei interne pot fi specificate prin denumiri simbolice alese de utilizator, denumiri care sugerează semnificația conținutului locațiilor respective.

De exemplu, locația 185, în care se va păstra numărul x , poate fi notată prin X , locația 213 pentru numărul y se va nota prin Y , locația 200, în care va fi depusă suma $x + y$, se va nota prin S . În limbajul de asamblare fragmentul de program pentru adunarea numerelor x și y va avea forma:

```
INC X
ADU Y
MEM S.
```

În general, există o corespondență directă între scrierea instrucțiunii în limbajul de asamblare și scrierea în limbajul cod calculator, ceea ce face ușoară traducerea (traducerea) limbajului de asamblare în limbajul cod calculator.

Traducerea constă în înlocuirea mnemonicii instrucțiunii și a adreselor simbolice prin șirurile binare respective. Această înlocuire este făcută de un program special, denumit program de asamblare sau asamblor.

Limbajele cod calculator și de asamblare sînt **limbaje dependente de calculator**. Această dependență constă în faptul că formatul, codurile și mnemonica instrucțiunilor exprimă structura internă a calculatorului. Programele elaborate în aceste limbaje sînt cele mai scurte și rapide, însă procesul de programare necesită un mare volum de muncă. Simplificarea procesului de programare se asigură prin utilizarea limbajelor independente de calculator (*FORTRAN*,

BASIC, PASCAL, C etc.), în care operațiile de prelucrare și tipurile de date nu sînt legate de echipamentele calculatorului. Însă, cu regret, detașarea utilizatorului de structura internă a calculatorului diminuează eficacitatea programelor respective.

Întrebări și exerciții

- ❶ Care este diferența dintre limbajul cod calculator și limbajul de asamblare?
- ❷ Cum se exprimă codurile instrucțiunilor și adresele locațiilor într-un limbaj de asamblare?
- ❸ Care este destinația și cum se realizează traducerea programelor scrise într-un limbaj de asamblare?
- ❹ Se consideră că denumirea simbolică *X* semnifică locația 100, denumirea *Y* – locația 101, iar denumirea *S* – locația 102. Exprimați în limbajul de asamblare (vezi *tabelul 6.1*) următoarele programe:

a) 01 100
04 101
02 102
02 100

b) 01 100
02 100
03 100

c) 01 101
04 100
02 102

d) 01 101
03 101
03 101
03 101
02 102

e) 01 100
02 101
03 101
02 100

f) 01 100
02 102
01 101
02 100

Care va fi conținutul locațiilor 100, 101 și 102 pînă și după execuția fiecărui program?

- ❺ Se consideră că denumirile simbolice *X*, *Y* și *S* specifică, respectiv, locațiile 100, 200 și 300. Traducați următoarele programe scrise în limbajul de asamblare:

a) INC X
SCD Y
MEM S
INC Y

b) INC X
INC X
SLTC S
STP

c) INC Y
ADU X
MEM S
STP

d) INC Y
ADU Y
ADU Y
ADU Y
MEM S

e) INC X
MEM S
INC Y
MEM X
INC X

f) INC X
MEM S
INC Y
MEM X
INC S

Explicați cum va fi executat fiecare program.

- ❻ Care este diferența dintre limbajele dependente și limbajele independente de calculator? Care sînt avantajele și dezavantajele fiecărui limbaj?

6.5. Resursele tehnice și resursele programate ale calculatorului

Numărul total al instrucțiunilor unui calculator depinde, în primul rînd, de capacitatea lui. În cazul unui sistem mare de calcul acest număr poate depăși 1000, pe cînd la calculatoarele foarte mici el nu este mai mare decît 100. O anumită operație poate fi efectuată la unele calculatoare cu o singură instrucțiune, pe cînd aceeași operație, în alte calculatoare, pentru care nu există o instrucțiune specifică, este efectuată cu ajutorul unei succesiuni de alte instrucțiuni existente.

Operațiile efectuate intern de către componentele electronice ale calculatorului sînt cunoscute ca **operații implementate prin echipamente**, pe cînd operațiile efectuate cu ajutorul unei secvențe de instrucțiuni sînt cunoscute ca **operații implementate prin program**. De exemplu, operația de extragere a rădăcinii pătrate într-un anumit calculator poate fi efectuată prin echipament electronic, în alt calculator — printr-un subprogram. Delimitarea dintre implementarea prin echipamente și program depinde de calculator. În *figura 6.4* este prezentată o astfel de delimitare în cazul unui calculator cu posibilități medii.

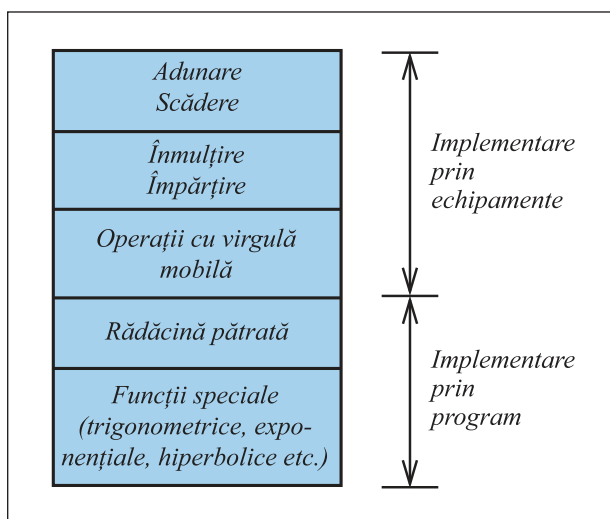


Fig. 6.4. Delimitarea între implementarea prin echipamente și prin program

Operațiile implementate prin echipamente se efectuează prin execuția unei singure instrucțiuni, pe cînd o operație implementată prin program necesită execuția mai multor instrucțiuni. Prin urmare, operațiile implementate prin echipamente se efectuează mai rapid, însă calculatorul respectiv este mai complex și, în consecință, mai scump. Din contra, operațiile implementate prin program se efectuează mai lent, însă calculatoarele respective sînt mai simple și, evident, mai ieftine.

Din analiza principiilor de funcționare a procesorului rezultă că toate echipamentele unui calculator devin inutile în absența programelor care guvernează efectuarea operațiilor necesare pentru rezolvarea unei probleme. Exact în același mod, programele sînt fără niciun folos în absența echipamentelor numerice res-

pective. Prin urmare, utilizarea tehnicii de calcul este posibilă numai în prezența atât a echipamentelor, denumite **resurse tehnice**, cât și a programelor respective, denumite **resurse programate**.

Resursele tehnice ale unui sistem de calcul modern includ procesorul, memoria internă, unitățile de memorie externă, echipamentele de intrare-ieșire etc. **Resursele programate** vor include subprogramele care realizează operațiile implementate prin program, programele care simplifică accesul la unitățile de intrare-ieșire, asambloarele, editoarele de texte, compilatoarele limbajelor algoritmice și, evident, programele elaborate de fiecare utilizator.

Menționăm că în literatura de specialitate resursele tehnice uneori sînt denumite prin cuvîntul englez *hardware* („produse de metal”), iar resursele programate – prin cuvîntul *software* („produse moi”). Respectiv, implementarea prin echipamente se numește implementare prin *hardware*, iar implementare prin program – implementare prin *software*.

Întrebări și exerciții

- ❶ De ce depinde numărul total al instrucțiunilor unui calculator?
- ❷ Cum se efectuează operațiile în cazurile implementării prin echipamente și în cele ale implementării prin program?
- ❸ Analizînd repertoriul instrucțiunilor calculatorului la care lucrați dvs., determinați cum sînt implementate următoarele operații:
 - înmulțirea și împărțirea numerelor binare;
 - adunarea și scăderea numerelor cu virgulă mobilă;
 - înmulțirea și împărțirea numerelor cu virgulă mobilă;
 - extragerea rădăcinii pătrate;
 - calcularea funcțiilor trigonometrice.
- ❹ Care sînt avantajele și dezavantajele implementării prin echipamente? Dar ale implementării prin program?
- ❺ Numiți resursele tehnice și resursele programate ale unui sistem de calcul. Care sînt resursele respective în cazul calculatorului la care lucrați dvs.?

6.6. Memorii externe pe benzi și discuri magnetice

Principiul de funcționare a memoriilor examinate constă în înregistrarea informației pe un strat magnetic, aflat în mișcare. Stratul magnetic este depus pe un suport neutru, acesta fiind, de regulă, o bandă de material plastic sau un disc de aluminiu. În calitate de strat magnetic se folosește mai frecvent oxidul de fier sau pelicule metalice subțiri de cobalt-nichel cu depunere în vid.

Înregistrarea sau citirea informației se efectuează cu ajutorul unui cap magnetic, reprezentat în *figura 6.5*.

Capul constă dintr-un miez, de regulă din tole (foițe) de permalloy foarte subțiri (0,05 mm) și o înfășurare.

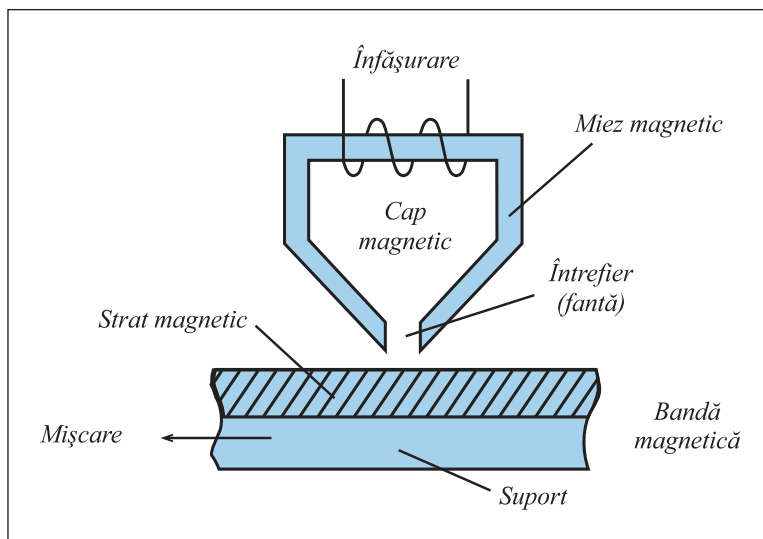


Fig. 6.5. Cap magnetic pentru înscrierea și citirea informației

Într-un strat magnetic neînregistrat, câmpurile magnetice ale particulelor de oxid de fier sînt orientate haotic, anulîndu-se reciproc. Pentru a **înregistra** cifra binară 0 sau 1, prin înfășurarea capului magnetic se trece impulsul respectiv de curent. Impulsul care traversează înfășurarea creează în întrefier un câmp magnetic intens care magnetizează stratul aflat în momentul actual sub cap. Direcția de magnetizare, deci informația binară înscrisă, depinde de direcția curentului în înfășurarea capului magnetic. În *figura 6.6* este prezentat un exemplu de înregistrare a informației binare 101101 pe un strat magnetic aflat în mișcare.

Distanța b determină lungimea porțiunii necesare pentru a memora o cifră binară. Valoarea lui b depinde de viteza de mișcare a suportului, de proprietățile fizice ale stratului magnetic, de elementele constructive ale capului magnetic etc.

Numărul de elemente de memorie binară pe unitatea de lungime a suportului se numește densitatea de înregistrare a informației.

În cazul înregistrărilor magnetice densitatea este dată de mărimea $1/b$. Valorile practice variază după tipul dispozitivului de memorare și firma constructoare, fiind de ordinul sutelor și miilor de biți pe milimetru de lungime a suportului.

În timpul operației de **citire** câmpul magnetic al particulelor de oxid de fier, trecînd prin dreptul întrefierului (*fig. 6.5*), induce în înfășurarea capului magnetic un semnal de ordinul 10^{-3} volți. Acest semnal este amplificat și transformat în semnal-standard care reprezintă cifra binară respectivă 0 sau 1.

În marea lor majoritate **memoriile externe pe bandă magnetică** reprezintă echipamente periferice autonome care transferă informația spre/de la memoria internă după receptarea comenzilor corespunzătoare de la procesorul calculatorului. O unitate de memorie pe bandă magnetică (*fig. 6.7*) include mecanismul de antrenare a benzii, dispozitivul de scriere-citire și circuitele de comandă aferente.

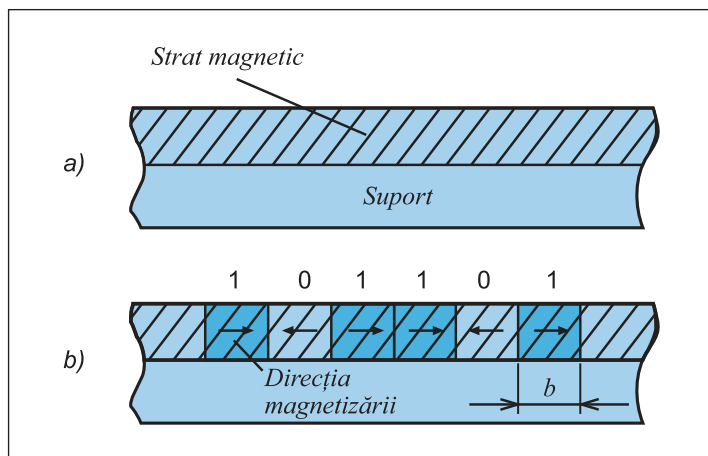


Fig. 6.6. Starea stratului magnetic pînă (a) și după înregistrare (b)

O operație de citire sau scriere se realizează în timpul deplasării benzii. Între două operații succesive banda este oprită. Evident, informațiile înregistrate pot fi citite numai în ordinea amplasării lor fizice pe bandă. Datorită acestui fapt, unitățile de bandă magnetică sînt denumite unități de memorie externă cu **acces secvențial**.

Timpul necesar pentru selectarea unei informații din multitudinea datelor memorate pe un suport se numește timp de acces.

Timpul de acces al unității de bandă magnetică depinde de viteza benzii și de locul amplasării informației de citit: la începutul, la sfîrșitul sau la mijlocul benzii. În cazul informațiilor înregistrate la sfîrșitul benzii, timpul de acces este de ordinul minutelor.

Capacitatea de memorare a unei benzi magnetice depinde de densitatea înregistrării, numărul de piste, lungimea benzii și are valori de ordinul 10^8 octeți.

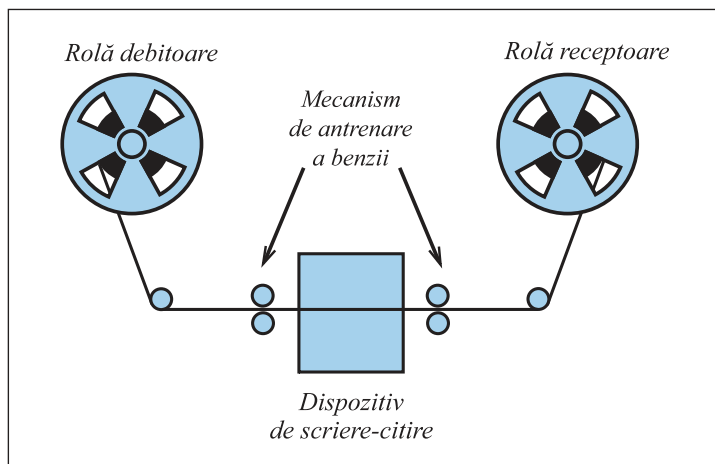


Fig. 6.7. Unitate de bandă magnetică

Din cauza timpului mare de acces și capacitatea de memorie relativ mică, benzile magnetice se utilizează, în general, numai pentru arhivarea informației.

Unitatea de discuri magnetice reprezintă la etapa actuală cea mai răspândită memorie externă a calculatoarelor numerice. Suportul informației este format dintr-un pachet de discuri, care poate fi fix sau amovibil, rotit cu o viteză de ordinul miilor de rotații pe minut. Discurile sînt acoperite cu un strat de material feromagnetic. Informația este înregistrată pe ambele fețe de-a lungul unor piste concentrice. Pentru aceasta fiecare față este explorată în plan orizontal de către un cap magnetic mobil (*fig. 6.8*).

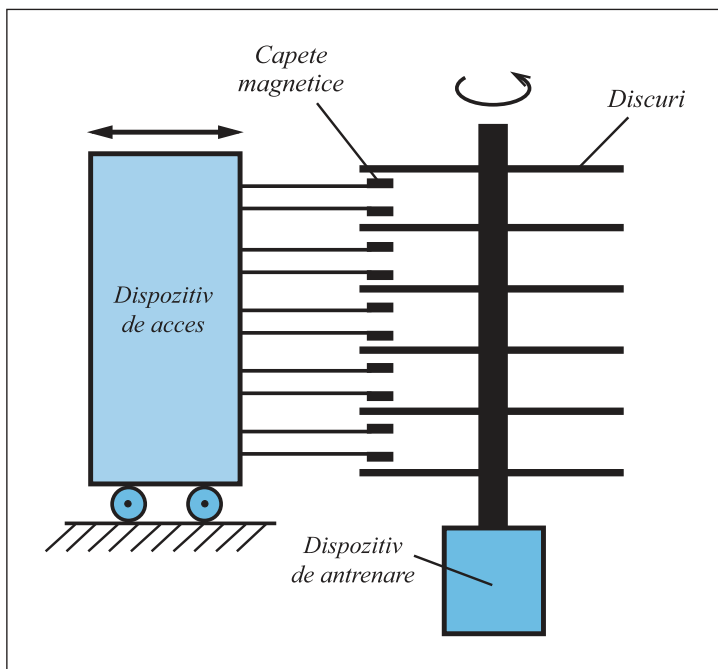


Fig. 6.8. Unitate de discuri cu capete mobile

Capetele sînt montate pe un braț extensibil și acționate de un mecanism pentru a ajunge în dreptul pistei selectate. Toate capetele unei unități de memorie cu discuri sînt poziționate simultan.

Timpul de acces al unității de discuri se compune din timpul necesar deplasării ansamblului de capete magnetice de pe cilindrul curent pe cilindrul indicat și din timpul necesar ca sectorul respectiv al discului să ajungă în dreptul capului magnetic. În practică se utilizează **timpul mediu de acces**, care pentru unitățile de disc moderne este de ordinul 10^{-3} secunde.

Amintim că în calculatoarele de performanță se utilizează unități de discuri cu capete magnetice fixe – cîte un cap magnetic pentru fiecare pistă. Aceste unități asigură un timp de acces de ordinul 10^{-4} secunde, însă sînt foarte scumpe.

Pentru schimbul de informații între calculatoare se utilizează discuri singulare din material plastic, denumite **discuri flexibile** sau **dischete**. Aceste discuri sînt încorporate într-o casetă din plastic sau în plicuri speciale. Organizarea fizică a

informației este aceeași ca și în cazul pachetelor de discuri, însă unitățile respective sînt mult mai simple și, evident, mai ieftine. Pentru a le deosebi de discurile flexibile, discurile convenționale ale calculatoarelor personale sînt denumite **discuri rigide, hard diskuri** sau **wincestere**.

Capacitatea de memorare a discurilor magnetice depinde de numărul de suprafețe ale pachetului, numărul de cilindri și densitatea de înregistrare. La momentul actual este obținută capacitatea de ordinul 10^{12} octeți pentru un pachet de discuri.

Întrebări și exerciții

- ❶ Cum sînt reprezentate cifrele binare 0 și 1 în înregistrările magnetice?
- ❷ Care este destinația capului magnetic?
- ❸ De ce depinde densitatea de înregistrare magnetică a informației?
- ❹ Cum se citește informația înregistrată pe un strat magnetic?
- ❺ Explicați cum funcționează unitatea de memorie pe bandă magnetică din *figura 6.7*.
- ❻ De ce depinde capacitatea de memorare a unei benzi magnetice?
- ❼ O bandă magnetică are lungimea de 750 de metri. Înregistrarea informației se efectuează pe 8 piste plus pista bitului de paritate. Capacitatea de memorare a benzii este 47 *Megaocteți*. Determinați densitatea de înregistrare.
- ❽ Viteza benzii magnetice este 2 m/s. În rola debitoare (*fig. 6.7*) sînt 750 m de bandă. Determinați timpul de acces la informația de la mijlocul benzii.
- ❾ Cum funcționează o unitate de discuri cu capete mobile?
- ❿ Cum este organizată informația pe un pachet de discuri magnetice?
- ⓫ Care este diferența dintre memoriile externe cu acces direct și acces secvențial?
- ⓬ De ce depinde timpul de acces al unității de discuri magnetice?
- ⓭ Determinați capacitatea discului flexibil cu care lucrați dvs.
- ⓮ Pentru unitatea de disc rigid cu care lucrați dvs. determinați:
 - capacitatea discului;
 - timpul mediu de acces.

6.7. Memorii externe pe discuri optice

Principiul de funcționare a **memoriilor pe discuri optice** constă în înregistrarea informației pe un strat reflectorizant aflat în mișcare. Stratul reflectorizant din aluminiu, aur sau argint este depus pe un suport transparent din masă plastică.

În funcție de modul de scriere și citire a informației, deosebim:

1) Discuri optice **numai pentru citire**. Informația pe astfel de discuri se înscrie de fabricant și nu poate fi modificată de utilizator. Abrevierea engleză a acestor discuri este *CD-ROM (Compact Disc – Read Only Memory)*.

2) Discuri optice **inscriptibile**. Informația pe astfel de discuri se înscrie de utilizator o singură dată, în continuare discul fiind disponibil numai pentru citire. Abrevierea engleză a acestor discuri este *CD-R (Compact Disc – Recordable)*.

3) Discuri optice **reinscriptibile**. Discurile în cauză permit mai multe cicluri de scriere/ștergere a informației. Abrevierea acestor discuri este *CD-RW* (*Compact Disc – ReWritable*).

Pentru a asigura compatibilitatea unităților de citire, formatul datelor și dimensiunile discurilor optice sînt standardizate. În *figura 6.9* este reprezentată structura discului optic *numai pentru citire* destinat publicului larg.

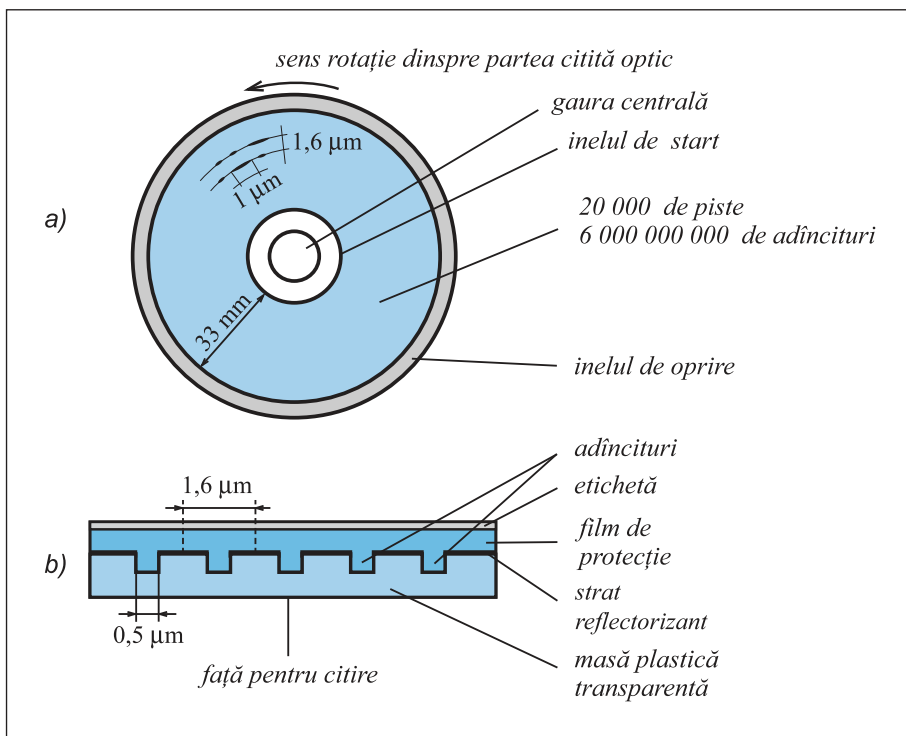


Fig. 6.9. Structura discului optic CD-ROM: a – poziționarea pistelor pe disc; b – secțiune a discului perpendiculară pe piste

Înregistrarea cifrelor binare pe astfel de discuri constă dintr-o succesiune de adâncituri (în limba engleză *pit*) realizate pe una dintre suprafețele discului. Aceste adâncituri sînt despărțite de mici pauze și sînt plasate pe suprafața discului sub forma unei piste în spirală.

Dimensiunile adânciturilor sînt de ordinul unui micron ($1 \text{ micron} = 10^{-3} \text{ milimetri}$), distanța dintre spire este de 1,6 micrometri, lungimea spiralei fiind de 5 300 de metri. Discul conține 20 000 de piste (spire) pe care se află circa $6 \cdot 10^9$ adâncituri. Capacitatea de memorare a discului este de 640 Megaocteți.

Citirea discului optic se realizează cu ajutorul unui fascicul de lumină care, după ce se reflectă de suprafața activă, este interceptat de o celulă fotosensibilă (*fig. 6.10*).

Parcurgînd pistele respective, fasciculul laser este reflectat cînd stratul reflectorizant se află în **punctul de focalizare** și nerefectat în caz contrar. Cu alte cuvinte, adânciturile de pe suprafața activă a discului optic schimbă (modulează) intensitatea fasciculului reflectat. În consecință, la ieșirea celulei fotosensibile se

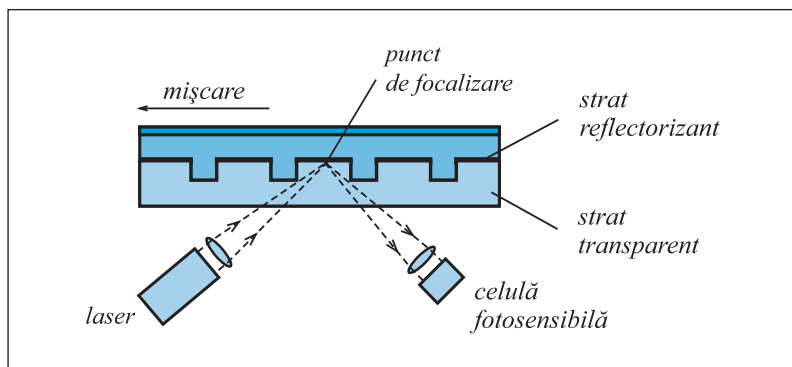


Fig. 6.10. Citirea discurilor optice

formează un semnal care redă succesiunea de cifre binare 0, 1 înregistrată pe disc la etapa fabricării.

În unitățile de disc optic moderne sursa de lumină – laserul și celula fotosensibilă – fotodioda se realizează într-un dispozitiv integral, denumit **cap optic de citire**. Viteza unghiulară a discului este de 200–600 de rotații pe minut, iar viteza liniară este de 1,4 m/s. În pofida vitezelor foarte mari, discul optic practic nu se uzează, întrucât între capul optic și disc nu există un contact mecanic direct. Prin urmare, durata de exploatare a unui disc *CD-ROM* este determinată de calitatea stratului reflectorizant și a stratului de protecție. În cazul utilizării aluminiului, din cauza oxidării, stratul reflectorizant se întunecă, reducând astfel viața unui disc optic la 10–15 ani. În cazul unui strat reflectorizant din aur numai o acțiune violentă care ar distruge stratul protector ar putea afecta calitatea unui disc optic.

De regulă, discurile optice *CD-ROM* se utilizează pentru tirajarea sistemelor de operare, mediilor de programare, enciclopediilor, jocurilor electronice și a altor informații destinate unui număr foarte mare de utilizatori.

Structura **discurilor optice inscriptibile și reinscriptibile** este redată în figura 6.11.

Din figură se observă că discurile examinate conțin un strat special, denumit **strat de înregistrare**. În cazul **discurilor inscriptibile** acest strat este format dintr-un material organic – *cyanin* sau *phtalocyanin* care se întunecă la încălzire.

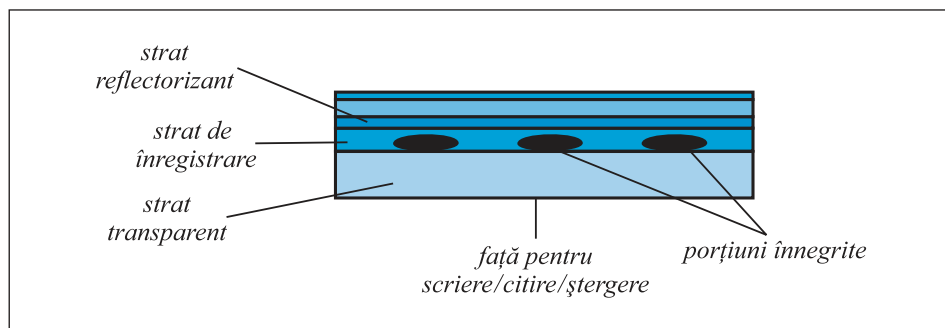


Fig. 6.11. Structura discurilor optice inscriptibile și reinscriptibile

Înregistrarea informației se realizează cu ajutorul unui fascicul laser foarte puternic care înnegrește porțiunile respective ale stratului de înregistrare.

La citire, porțiunile întunecate blochează trecerea fascicului laser spre stratul reflectorizant, modificînd astfel intensitatea luminii captate de celula fotosensibilă (fig. 6.10). Întrucît fasciculul laser folosit la citire este foarte slab, discul inscriptibil poate fi citit de mai multe ori fără a distruge informația înregistrată.

În cazul **discurilor reinscriptibile** materialul din care este format stratul de înregistrare devine din nou transparent dacă este încălzit pînă la o temperatură specială, denumită **temperatură critică**. Evident, după ștergere, pe discul reinscriptibil pot fi înscrise date binare noi. Discurile moderne *CD-RW* suportă pînă la 10 000 de cicluri de scriere/ștergere.

Neajunsul comun al discurilor inscriptibile și reinscriptibile este faptul că ele sînt mult mai sensibile la temperatură, durata lor de exploatare fiind mai mică decît a discurilor *CD-ROM*. Aceste discuri sînt și mai scumpe, întrucît pentru o citire sigură stratul reflectorizant este realizat din aur sau argint.

De obicei, discurile optice *CD-R* și *CD-RW* se utilizează pentru difuzarea informațiilor operative unui cerc bine determinat de utilizatori – baze de date, pachete specializate de programe, rapoartele unor congrese importante, tablourile unei expoziții, documente multimedia etc.

Menționăm faptul că realizările tehnologice din ultimul deceniu au condus la apariția discurilor optice de tipul *DVD* (*Digital Versatile Disc* – Disc Digital Multifuncțional). Capacitatea unui disc *DVD* este de aproape șapte ori mai mare decît a unui disc de tipul *CD*.

Întrebări și exerciții

- 1 Cum sînt reprezentate cifrele binare 0, 1 în înregistrările optice?
- 2 Care este capacitatea de memorare a unui disc optic?
- 3 Cum se înregistrează informația pe un disc optic *CD-ROM*?
- 4 De ce depinde capacitatea de memorare a unui disc optic?
- 5 E cunoscut faptul că lungimea spiralei pe care se înregistrează informația unui disc optic este de 5300 m. Capacitatea de memorare a discului este de 640 *Mega-octeți*. Determinați densitatea de înregistrare a informației pe discul optic.
- 6 Care este domeniul de utilizare a discurilor *CD-ROM*?
- 7 De ce depinde timpul de acces la informațiile de pe un disc optic?
- 8 Viteza liniară a discului optic este de 1,4 m/s. Cunoscînd densitatea de înregistrare a informației $d = 128$ *Kocteți/m*, determinați **viteza de transmisie a datelor** de la unitatea de disc la unitatea centrală. Amintim că viteza de transmisie a datelor se măsoară în *biți*, *Kbiți* sau *Mbiți* pe secundă.
- 9 Pe cele 20 000 de piste (spire) ale unui disc optic sînt înregistrați circa 640 de *Mega-octeți*. Cîtă informație conține o singură pistă a discului optic?
- 10 Pe un disc *CD-ROM* sînt înregistrate în formă binară circa 74 de minute de muzică. Determinați cîtă informație va conține un cîntec cu durata de 4 minute și 30 de secunde. Cîte piste va ocupa cîntecul respectiv?

- 11 Cum se citește informația de pe un disc optic? Care este destinația capului optic de citire?
- 12 Explicați destinația straturilor unui disc optic inscriptibil. Cum se realizează înregistrarea informației pe discul în cauză?
- 13 De ce depinde durata de exploatare a unui disc *CD-ROM*? Dar a discurilor *CD-R* și *CD-RW*?
- 14 Cum se înregistrează și se șterge informația de pe un disc optic reinscriptibil?
- 15 Care este domeniul de utilizare a discurilor *CD-R* și *CD-RW*?
- 16 Aflați parametrii tehnici ai unității de disc optic cu care este dotat calculatorul dvs. Memorizați regulile de exploatare a discurilor și a unităților de memorie externă pe discuri optice.

6.8. Vizualizatorul și tastatura

Vizualizatorul este un dispozitiv de ieșire prin intermediul căruia informația este prezentată utilizatorului pe ecranul unui tub catodic. Schema funcțională a vizualizatorului este prezentată în *figura 6.12*.

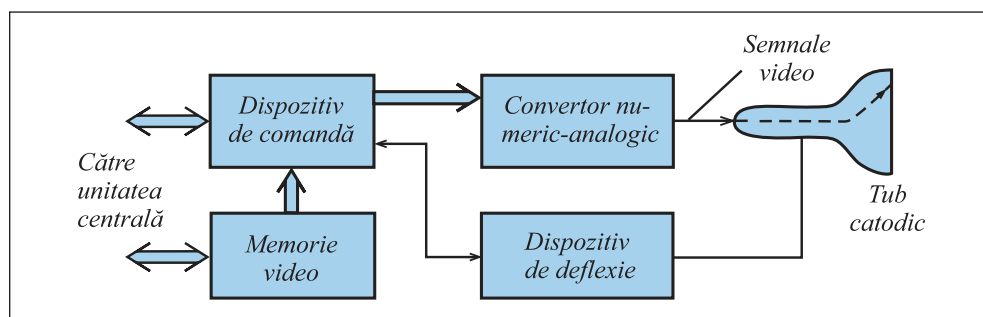


Fig. 6.12. Schema funcțională a vizualizatorului

Ca și în cazul televizoarelor pentru publicul larg, imaginea pe ecranul tubului catodic se formează din puncte. Punctele sînt activate de un fascicul de electroni. Culoarea și luminanța fiecărui punct este controlată de semnale video, aplicate la intrările corespunzătoare ale tubului catodic. Parcurgerea punctelor într-o ordine prestabilită, de regulă, pe linii de la stînga la dreapta și de sus în jos, este asigurată de dispozitivul de deflexie.

Fiecărui punct al ecranului îi corespunde o locație în memoria video a vizualizatorului. Locațiile conțin informații referitoare la culoarea și luminanța punctelor respective. Dispozitivul de comandă citește locațiile memoriei video în ordinea parcurgerii punctelor de pe ecran. Conținutul fiecărei locații este interpretat drept o instrucțiune referitoare la modul de activare a punctului care îi corespunde. Semnalele video, necesare pentru comanda fasciculului de electroni, sînt formate de convertoare numeric-analogice.

Memoria video a vizualizatorului este încărcată de unitatea centrală a calculatorului. În orice moment, calculatorul poate modifica conținutul acestei memo-

rii. În consecință, se va schimba și imaginea afișată pe ecran. Imaginile pot fi animate prin modificarea conținutului memoriei video la frecvențe utilizate în cinematografie.

Vizualizatorul poate funcționa în unul dintre cele două regimuri: alfanumeric sau grafic.

În **regim alfanumeric** ecranul este împărțit în zone convenționale, numite **zone-caracter**. De regulă, aceste zone formează 25 de linii cu 80 de caractere pe linie. În fiecare zonă poate fi afișat un singur caracter dintr-un set de 256 de caractere. Setul de caractere este constituit din literele mari și mici ale alfabetului latin, cifrele zecimale, simbolurile matematice, semnele de punctuație și unele caractere semigrafice, utilizate la afișarea pe ecran a tabelelor, diagramelor, che-narelor etc. Fiecare zonă-caracter poate avea o culoare pentru caracter și altă culoare pentru fundal, ceea ce îi permite utilizatorului să afișeze pe ecran texte cu litere de diferite culori.

În **regim grafic** utilizatorul poate controla afișarea pe ecran a fiecărui punct. Numărul de puncte pe orizontală și verticală determină rezoluția vizualizatorului. De exemplu, expresia „rezoluția 640×200” înseamnă că vizualizatorul are 640 de puncte pe orizontală și 200 pe verticală.

Există mai multe norme internaționale care reglementează rezoluția și numărul de culori ale vizualizatoarelor. Aceste caracteristici sînt recunoscute și respectate de firmele producătoare.

De exemplu, în cazul calculatoarelor personale cele mai răspîndite sînt normele denumite *EGA*, *VGA* și *SVGA*. Norma *EGA* stipulează că vizualizatorul are o rezoluție de 640×350 de puncte, permițînd reprezentarea a 64 de culori.

Norma *VGA*, păstrînd compatibilitatea cu *EGA*, oferă ca facilități suplimentare rezoluția 640×480 de puncte și 256 de culori distincte.

În scopul îmbunătățirii calității imaginii redată, norma *SVGA* adaugă rezoluția suplimentară de 1024×768 de puncte.

Menționăm că realizările tehnologice din ultimii ani au permis înlocuirea tuburilor catodice cu ecrane plate, fapt ce asigură o reducere substanțială a dimensiunilor vizualizatorului și o îmbunătățire semnificativă a rezoluției imaginilor afișate. Astfel de vizualizatoare se caracterizează printr-un consum redus de energie și rezoluții de 1920×1200 de puncte.

Tastatura este un dispozitiv de intrare care transformă acționarea unei taste într-un cuvînt binar, accesibil echipamentelor calculatorului.

Partea electronică a unei tastaturi constă dintr-un codificator. La intrările codificatorului se aplică semnalele logice provenite de la taste. La ieșire se furnizează cuvintele unui cod binar, de obicei standard (*ISO*, *ACSII* etc.). Unele tipuri de tastaturi pot fi prevăzute cu un generator audio, care la acționarea tastelor produce un sunet specific.

La noi, ca și în țările anglofone, cea mai răspîndită este tastatura de tipul *QWERTY*, denumirea căreia provine de la amplasarea caracterelor *Q*, *W*, *E*, *R*, *T* și *Y* în rîndul de sus al tastelor alfanumerice. În țările francophone se utilizează tastatura *AZERTY*, în Germania *QWERTZ* etc.

Deși locul amplasării tastelor și numărul lor pot să difere, destinația tastelor este aceeași.

Tastatura dispune de următoarele grupe de taste: alfanumerice, funcționale și speciale. Grupul de taste alfanumerice include tastele cifrelor zecimale, tastele caracterelor alfabetului englez, tastele simbolurilor matematice și ale semnelor de punctuație. Grupul de taste funcționale include tastele <F1>, <F2>, ... , <F12>. Tastele respective nu au o destinație prestabilită și semnificația lor este definită de programul care derulează pe calculator. Tastele speciale se utilizează pentru poziționarea cursorului vizualizatorului, pentru introducerea în calculator a unor cuvinte binare care nu au taste proprii etc.

În calculatoarele de performanță, vizualizatorul și tastatura pot forma un echipament periferic unitar, denumit **consolă**. Consola utilizată de operator pentru dirijarea proceselor de calcul se numește **monitor**.

Întrebări și exerciții

- ❶ Explicați cum funcționează vizualizatorul. Care este destinația părților componente ale unui vizualizator?
- ❷ Cum poate fi schimbată imaginea afișată pe ecranul unui vizualizator? Cum pot fi animate imaginile de pe ecran?
- ❸ Prin ce se deosebesc regimurile de funcționare a vizualizatorului?
- ❹ Care sînt indicii principali de calitate ai unui vizualizator?
- ❺ Determinați tipul vizualizatorului la care lucrați dvs. Aflați rezoluția și numărul de culori disponibile.
- ❻ Care sînt părțile componente ale unei tastaturi? Cum se determină tipul tastaturii?
- ❼ Numiți grupurile de taste și destinația lor.

6.9. Imprimantele

Imprimantele sînt dispozitive de ieșire care furnizează rezultatele sub forma unui document tipărit. În funcție de **tehnica de tipărire** utilizată, imprimantele se pot clasifica în:

- imprimante mecanice, în care tipărirea se face prin acționarea unor ciocănașe sau ace;
- imprimantele laser, în care tipărirea se face folosind metode electrostatice ca la mașinile de copiat;
- imprimantele cu jet de cerneală;
- imprimantele termice, funcționarea cărora se bazează pe utilizarea unei hîrtii speciale care își schimbă culoarea în funcție de temperatură.

Principiul de funcționare a **imprimantei matriciale cu ace** este prezentat în *figura 6.13*.

Capul de imprimare conține un grup de ace metalice subțiri care la momentul potrivit lovesc prin bandă tușată în foaia de hîrtie. Configurația ácelor ce lovesc în fiecare moment și avansarea capului de-a lungul liniei de imprimat determină imaginile tipărite.

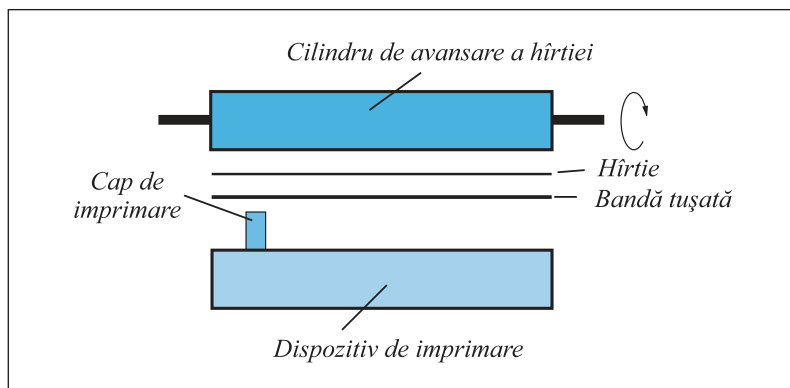


Fig. 6.13. Principiul de funcționare a imprimantei matriciale cu ace

Imprimantele matriciale pot funcționa în regim grafic și regim alfanumeric.

În **regim grafic**, calculatorul comandă tipărirea pe hîrtie a fiecărui punct aparate. Evident, din puncte pot fi formate orice imagini: grafice, desene tehnice, caractere cu configurații concepute de utilizator. Imprimarea informației în regim grafic este lentă din cauza deplasărilor multiple ale capului de imprimare și a caracterului discontinuu al avansului de hîrtie cu pași foarte mici.

În **regim alfanumeric**, calculatorul transmite imprimantei numai codurile caracterelor de tipărit. Fiecărui cod îi corespunde o imagine din puncte care se păstrează într-o memorie specială a imprimantei. Imaginile respective sînt tipărite prin una sau, cel mult, două-trei treceri ale capului de imprimare.

Imprimarea în regim alfanumeric este mult mai rapidă, însă pot fi tipărite numai caractere, imaginile cărora au fost, în prealabil, înmagazinate în memoria imprimantei. Imprimantele matriciale simple au cîteva seturi de caractere, înscrise într-o memorie permanentă. Imprimantele matriciale mai performante permit încărcarea prin program a mai multe seturi de caractere cu configurații concepute de utilizator.

E cazul să subliniem că cu cît numărul ácelor din capul de imprimare este mai mare, cu atît calitatea imprimării este mai bună. În prezent se utilizează imprimante cu 9 sau cu 24 de ace. Calitatea imprimării poate fi îmbunătățită prin tipărirea repetată (de 2-4 ori) a aceluiași caracter pe același loc. Viteza de tipărire a imprimantelor mecanice cu ace este de 150–500 de caractere pe minut.

Principiul de funcționare a **imprimantelor laser** este prezentat în *figura 6.14*.

Elementul principal al acestei imprimante este un tambur acoperit cu un strat semiconductor care își poate schimba proprietățile electrice sub acțiunea luminii. Mai întîi, suprafața tamburului se electrizează. Cu ajutorul razelor laser pe suprafața electrizată a tamburului se proiectează punctele imaginii de tipărit. Întrucît sectoarele care au fost luminate își schimbă conductibilitatea, sarcinile electrice respective se neutralizează. Prin urmare, pe suprafața tamburului se formează o imagine electrică ascunsă. Developarea imaginii se efectuează cu ajutorul unor particule de pulbere colorată, atrase de sectoarele electrizate ale tamburului. Imaginile de pe tambur se transferă pe hîrtie și se fixează prin topirea particulelor de pulbere.

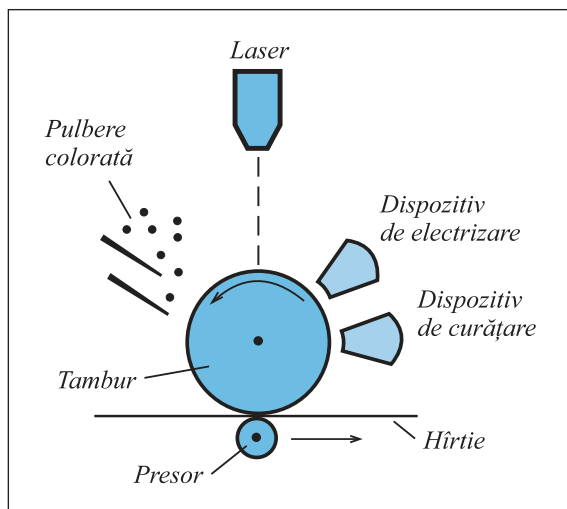


Fig. 6.14. Principiul de funcționare a imprimantei laser

În continuare, toate sarcinile electrice de pe suprafața tamburului sînt neutralizate, iar resturile de pulbere sînt curățate.

Imprimantele laser sînt cele mai bune dintre imprimantele moderne. Textele și grafica tipărite la aceste imprimante nu se deosebesc de cele tipografice. Viteza de tipărire este de 5-15 pagini pe minut.

Imprimantele cu jet de cerneală formează punctele unei imagini din picături microscopice pulverizate pe hîrtie de niște ajutaje speciale. Ele asigură o calitate foarte bună a tiparului și sînt utilizate pentru imprimarea color. Aceste imprimante sînt mai scumpe decît imprimantele cu ace și necesită o îngrijire tehnică deosebită.

Imprimantele termice utilizează o matrice de ace a căror încălzire selectivă este determinată în funcție de caracterul ce urmează a fi tipărit. Viteza de imprimare este relativ mică, circa 300 de rînduri pe minut, însă avantajul principal constă în dimensiunile reduse ale imprimantei.

Întrebări și exerciții

- ❶ Cum se clasifică imprimantele în funcție de tehnica de tipărire?
- ❷ Care sînt părțile componente ale unei imprimante?
- ❸ Care este principiul de funcționare a imprimantei matriciale cu ace? Cum funcționează această imprimantă?
- ❹ Explicați cum funcționează o imprimantă laser. Care este avantajul principal al acestei imprimante?
- ❺ Determinați tipul imprimantei cu care lucrați dvs. Aflați parametrii tehnici ai imprimantei: setul de caractere, regimurile de funcționare, capacitatea memoriei-tampon, viteza de imprimare.

6.10. Clasificarea calculatoarelor

Caracteristica generală a unui calculator include următoarele date:

- viteza de operare;
- capacitatea memoriei interne;
- componența, capacitatea și timpul de acces ale unităților de memorie externă;
- componența și parametrii tehnici respectivi ai echipamentelor periferice;
- parametrii de masă și gabarit;
- costul.

În funcție de aceste date, calculatoarele moderne se clasifică în 4 categorii:

- supercalculatoare;
- calculatoare mari (macrocalculatoare);
- minicalculatoare;
- microcalculatoare (calculatoare personale).

Supercalculatoarele pot executa peste 10^{15} (1000 bilioane) de operații pe secundă, iar prețul lor depășește 20 de milioane de dolari. Cercetări și proiectări în industria supercalculatoarelor se realizează în SUA și Japonia de firmele *IBM*, *Gray Research*, *Fujitsu*, *ETA Systems*, *Sutherland* etc. Supercalculatoare se utilizează în prelucrări extrem de complexe ale datelor în aeronautică, fizica nucleară, astronomică, seismologie, prognoza meteo etc.

Calculatoarele mari (în engleză *mainframe* „dulapul principal”) pot executa sute de bilioane de operații pe secundă ($1 \text{ bilion} = 10^{12}$), prețul variind între 20 de mii și câteva milioane de dolari. Schema-bloc a calculatoarelor date este prezentată în *figura 6.2*. De regulă, calculatoarele mari includ zeci de unități de disc magnetic și imprimante, sute de console aflate la diferite distanțe de unitatea centrală. Aceste calculatoare se utilizează în cadrul unor mari centre de calcul și funcționează în regim non-stop. Principalele firme producătoare de calculatoare mari sînt *IBM*, *Hitachi*, *Amdahl*, *Fujitsu* etc.

Minicalculatoarele efectuau sute de milioane de operații pe secundă, iar prețul lor nu depășea 200–300 mii de dolari. Echipamentele periferice ale unui minicalculator includeau câteva discuri magnetice, una sau două imprimante, mai multe console. Minicalculatoarele erau mai ușor de utilizat și operat decît calculatoarele mari și se utilizau în proiectarea asistată de calculator, în automatizări industriale, pentru prelucrarea datelor în experimentele științifice etc. Dintre firmele ce produceau minicalculatoare vom remarca *IBM*, *Wang*, *Texas Instruments*, *Data General*, *DEC*, *Hewlett-Packard* etc. În prezent minicalculatoarele au fost înlocuite de calculatoarele personale.

Microcalculatoarele, denumite și calculatoare personale, sînt realizate la prețuri scăzute – între 100 și 15 000 de dolari și asigură o viteză de calcul de ordinul miliardelor de operații pe secundă. Schema-bloc a unui calculator personal este prezentată în *figura 6.2*.

De obicei, echipamentele periferice ale unui microcalculator includ o unitate de disc rigid, o unitate de disc flexibil, o unitate de disc optic, o imprimantă și o consolă. Structura modulară și gruparea tuturor echipamentelor în jurul unei magistrale permite configurarea microcalculatorului în funcție de necesitățile individuale ale fiecărui utilizator.

Corporații care produc microcalculatoare există în foarte multe țări, însă lideri mondiali, unanim recunoscuți, sînt firmele *IBM*, *Apple*, *Hewlett-Packard*, *Dell* etc.

Întrebări și exerciții

- 1 Numiți parametrii mai semnificativi ai unui calculator. Determinați parametrii respectivi ai calculatorului cu care lucrați dvs.
- 2 Cum se clasifică calculatoarele în funcție de parametrii tehnici și economici?
- 3 Dați o caracteristică succintă a fiecărei categorii de calculatoare: supercalculatoare, calculatoare mari, minicalculatoare și microcalculatoare.
- 4 Folosind un server de căutare, găsiți în Internet informații detaliate despre principalii producători de calculatoare personale.

6.11. Microprocesorul

Microprocesorul este un circuit integrat care implementează funcțiile unității centrale de prelucrare a informației, și anume – extragerea și executarea instrucțiunilor.

De regulă, un microprocesor conține un dispozitiv aritmetic și altul de comandă, un grup de registre, destinate păstrării temporale a datelor frecvent utilizate, magistralele și circuitele de comandă aferente (fig. 6.15).

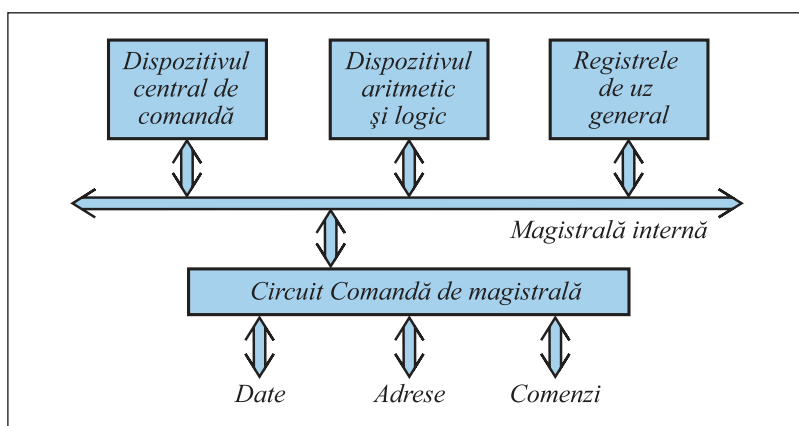


Fig. 6.15. Schema funcțională a unui microprocesor

Microprocesorul interacționează cu unitățile de memorie și echipamentele periferice prin intermediul a trei magistrale: *Date*, *Adrese* și *Comenzi*. Traficul de informații prin magistrale este controlat de circuitul *Comandă de magistrală*.

Extragerea și executarea instrucțiunilor are loc sub controlul **dispozitivului central de comandă**. Pentru aceasta, prin magistrala de adrese se indică adresa locației memoriei interne, iar prin magistrala de comenzi se transmit semnalele necesare de scriere sau citire. Datele citite sau scrise se transmit prin magistrala de date.

Microprocesoarele se apreciază după următorii parametri:

- lungimea cuvântului;
- frecvența ceasului de sistem;
- capacitatea magistralelor.

Lungimea cuvîntului reprezintă numărul de biți ai succesiunilor binare care pot fi memorizate în registre și prelucrate de dispozitivul aritmetic al microprocesorului. Microprocesoarele moderne au lungimea cuvîntului de 32, 64 și mai mulți biți.

Frecvența ceasului de sistem reprezintă numărul de impulsuri pe secundă pe care le produce generatorul de impulsuri de tact din componența dispozitivului de comandă. Frecvența ceasului de sistem se măsoară în **Megahertz** ($1\text{ MHz} = 10^6\text{ Hz}$). Întrucît impulsurile de tact sincronizează executarea microoperațiilor în toate dispozitivele microprocesorului, frecvența ceasului de sistem caracterizează rapiditatea microprocesorului.

Capacitatea magistralei reprezintă numărul de biți ai succesiunilor binare transmise prin magistrală. De obicei, capacitatea magistralei de date este egală sau mai mare decît lungimea cuvîntului microprocesorului. În caz contrar, pentru a transmite un cuvînt, sînt necesare mai multe cicluri ale magistralei de date.

Capacitatea magistralei de adrese determină spațiul de adrese care pot fi accesate direct de microprocesor. Astfel, un microprocesor avînd capacitatea magistralei de adrese de 16 biți poate accesa 2^{16} locații ale memoriei interne, iar un microprocesor cu capacitatea magistralei de adrese de 32 de biți poate accesa direct 2^{32} locații.

În ansamblu, lungimea cuvîntului, frecvența ceasului de sistem și capacitatea magistralelor determină **capacitatea de prelucrare a microprocesorului**, care se măsoară în **Mips** – Megainstrucțiuni pe secundă. Pentru exemplificare, în *tabelul 6.2* sînt prezentate caracteristicile principale ale microprocesoarelor din familia *Intel*.

Tipul microprocesorului și frecvența ceasului de sistem al calculatorului personal cu care lucrați dvs. poate fi aflat cu ajutorul meniului contextual al pictografei „My computer”. Amintim că acest meniu se afișează pe ecran cu ajutorul unui clic-dreapta pe pictograma respectivă.

Tabelul 6.2

**Caracteristicile principale
ale microprocesoarelor din familia *Intel***

Micro-procesor	Lungime cuvînt, <i>bit</i>	Frecvență, <i>MHz</i>	Capacitate de prelucrare, <i>Mips</i>
Pentium I	32	200	200
Pentium II	32	300	400
Pentium III	32	1400	1200
Pentium 4	32	3800	2500
Pentium D	64	3400	4000

Întrebări și exerciții

- ❶ Explicați destinația echipamentelor din componența unui microprocesor (*fig. 6.15*).
- ❷ Care sînt parametrii principali ai unui microprocesor? Explicați semnificația fiecărui parametru.
- ❸ Determinați tipul și parametrii principali ai microprocesorului din componența calculatorului personal cu care lucrați dvs.

Test de autoevaluare nr. 6

1. Indicați corespondența dintre denumirile unităților funcționale ale calculatorului (coloana din stânga) și destinația acestora (coloana din dreapta):

(1) dispozitivul de intrare;	(a) efectuarea operațiilor aritmetice și logice elementare;
(2) memoria;	(b) colorarea imaginilor și memorarea lor pe discuri optice;
(3) dispozitivul aritmetic și logic;	(c) extragerea datelor din calculator;
(4) dispozitivul de ieșire;	(d) furnizarea semnalelor de comandă necesare executării secvențiale a instrucțiunilor;
(5) dispozitivul central de comandă;	(e) introducerea documentelor în calculator și corectarea greșelilor gramaticale;
(6) procesorul.	(f) înmagazinarea datelor inițiale, intermediare și finale ale problemei, precum și a instrucțiunilor care indică secvența calculelor;
	(g) prelucrarea automată a informației conform programului înmagazinat în memorie;
	(h) efectuarea calculelor aritmetice și afișarea rezultatelor la ecran;
	(i) introducerea datelor din mediul exterior în calculator.

2. Numiți cel puțin cinci echipamente periferice pe care le cunoașteți dvs.

3. Care dintre afirmațiile ce urmează sînt adevărate:

- a) un calculator poate fi conceput fără ca să aibă o unitate de memorie externă;
- b) memoria internă este un dispozitiv periferic;
- c) memoria externă este mai lentă decît memoria internă;
- d) unitatea de disc magnetic este o memorie internă;
- e) un calculator poate fi conceput fără ca să aibă o unitate de memorie internă;
- f) memoria externă este un dispozitiv periferic?

4*. Numiți destinația magistralei din componența calculatorului personal.

5*. Desenați schema-bloc a calculatorului personal. Care componente sînt obligatorii și care opționale pentru funcționarea calculatorului?

6*. Este cunoscut faptul că instrucțiunile cu trei adrese conțin următoarele cîmpuri: *Codul instrucțiunii*, *Adresă operand 1*, *Adresă operand 2* și *Adresă rezultat*. Explicați destinația fiecărui cîmp.

* Numai pentru profilul real.

7*. Explicați cum vor fi executate următoarele instrucțiuni cu trei adrese:

a) 01 101 153 342;

b) 04 508 391 216;

c) 03 751 852 031;

d) 02 450 709 011.

Operațiile aritmetice și logice sînt codificate în felul următor: 01 – adunarea; 02 – scădere; 03 – înmulțirea; 04 – împărțirea.

8*. Indicați tipul instrucțiunilor ce urmează (operaționale, de transfer, de salt sau de intrare-ieșire):

a) adunarea a două numere;

b) scrierea unui șir de octeți pe discul magnetic;

c) deplasarea unui cuvînt binar de la dreapta spre stînga;

d) citirea unui șir de octeți de pe discul optic;

e) compararea a două numere și transferul controlului în funcție de rezultatul comparării;

f) înmulțirea a două numere;

g) încărcarea în registrul procesorului a unui număr din memoria internă.

9*. Se consideră că denumirea simbolică *X* semnifică locația 205, denumirea *Y* – locația 421, iar denumirea *S* – locația 783. Exprimați în limbajul de asamblare (vezi tabelul 6.1) următorul program:

```
01 205  
02 783  
04 421  
03 205.
```

10*. Se consideră că denumirile simbolice *X*, *Y* și *S* specifică, respectiv, locațiile 971, 583 și 461. Translați următoarele programe scrise în limbajul de asamblare (vezi tabelul 6.1):

```
INC Y  
MEM S  
INC X  
MEM Y  
INC S
```

11. Indicați tipul resurselor ce urmează (tehnice sau programate):

a) procesorul;

b) editorul de texte;

c) aplicația de calcul tabelar;

d) imprimanta;

e) aplicația de poștă electronică;

f) tastatura;

g) sistemul de operare;

h) vizualizatorul.

12. Care este diferența dintre memoriile externe cu acces direct și acces secvențial?

13. Care dintre afirmațiile ce urmează sînt adevărate:

- a) unitatea de memorie de bandă magnetică asigură accesul direct la înregistrări;
- b) timpul de acces al unității de bandă magnetică este mai mare ca cel al unității de discuri magnetice;
- c) unitatea de memorie de discuri magnetice asigură accesul secvențial la înregistrări;
- d) pe o bandă magnetică înregistrările sînt grupate pe cilindri;
- e) timpul de acces al unității de disc rigid este mai mic ca cel al unității de disc flexibil?

14. Care dintre următoarele afirmații sînt adevărate:

- a) toate discurile optice au dezavantajul că odată ce au fost scrise nu mai pot fi rescrise;
- b) în general, discurile magnetice rigide au capacități de stocare mai mari decît ale discurilor optice;
- c) capacitatea de stocare a unui disc optic depinde de viteza cu care el se rotește;
- d) un disc magnetic flexibil are o capacitate de stocare mai mare decît un disc optic;
- e) numărul ciclurilor de scriere/ștergere a discurilor optice reinscriptibile nu poate depăși o anumită limită?

15*. Viteza liniară a discului optic este de 5,6 m/s, iar densitatea de înregistrare a informației este de 512 *Kocteți*/m. Determinați viteza de transmisie a datelor de la unitatea de disc optic la unitatea centrală.

16. Alegeți din lista ce urmează parametrii ce caracterizează un vizualizator:

- a) rezoluția;
- b) numărul de pagini ce pot fi afișate într-o secundă;
- c) diagonala ecranului;
- d) numărul de culori pe care le poate afișa;
- e) viteza de imprimare.

17. Alegeți din lista ce urmează parametrii ce caracterizează o imprimantă:

- a) rezoluția;
- b) numărul de culori pe care le poate imprima;
- c) diagonala imprimantei;
- d) numărul de instrucțiuni pe secundă;
- e) viteza de imprimare.

18. Indicați parametrii de bază ce caracterizează un calculator.

19*. Indicați parametrii de bază ai microprocesorului.

7.1. Introducere în rețele

Odată cu extinderea domeniilor de aplicare a calculatoarelor, a crescut și numărul utilizatorilor ce doreau să aibă acces la mijloace eficiente de prelucrare și stocare a unor informații comune.

De exemplu, în cazul proiectării unei clădiri, mai mulți specialiști – arhitectul, inginerul, pompierul etc. doresc să aibă acces și să introducă, dacă e necesar, modificări în desenele tehnice ale construcției în curs de elaborare. În cazul unei companii de transporturi aeriene biletele la una și aceeași cursă pot fi vândute de mai multe agenții aflate în orașe diferite.

Soluția inițială la astfel de probleme a constituit-o conectarea la un calculator central, de mare capacitate, a mai multe terminale (fig. 7.1).

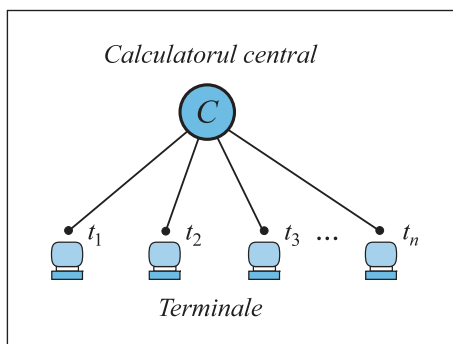


Fig. 7.1. Sistem centralizat de calcul

De regulă, un terminal constă dintr-un vizualizator, o tastatură și, dacă e necesar, o imprimantă. Neajunsul principal al unui sistem centralizat de prelucrare a datelor este fiabilitatea redusă și utilizarea inefficientă a resurselor de calcul.

Cu timpul, a apărut tendința de trecere de la sistemele centralizate la instalarea de calculatoare la fiecare utilizator și asigurarea unor legături de comunicație efficientă între ele (fig. 7.2).

Numim rețea de calculatoare o mulțime de calculatoare ce pot schimba informații prin intermediul unei structuri de comunicație.

Calculatoarele unei rețele se conectează la structura de comunicație prin intermediul unor unități de intrare-ieșire dedicate, numite **adaptoare de rețea**.

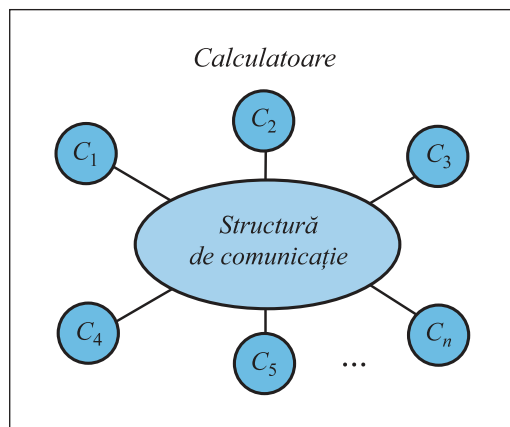


Fig. 7.2. Rețea de calculatoare

Evident, în cadrul unei rețele fiecare calculator, mai exact, fiecare adaptor de rețea, are o adresă unică, denumită **adresă de rețea**.

De exemplu, o rețea de calculatoare poate fi construită utilizând ca **structură de comunicație** rețeaua existentă de telefoane. În acest caz, adaptorul de rețea va include un modulator pentru conversiunea semnalelor digitale furnizate de calculator în semnale telefonice și un demodulator pentru operația inversă. Dispozitivul respectiv poartă denumirea de **modem (modulator-demodulator)**. Adresa de rețea este dată de numărul de telefon al postului la care este conectat modemul.

În general, o structură de comunicație este formată din **linii de transmisie** a semnalelor. Aceste linii pot fi:

- cabluri cu fire torsadate;
- cabluri coaxiale;
- cabluri optice;
- linii cu microunde (terestre sau prin satelit).

Cablurile cu fire torsadate sînt asemănătoare celor telefonice și asigură o capacitate de transmisie de pînă la 1 *Mbit/s*. **Cablurile coaxiale**, asemănătoare celor din rețelele de televiziune prin cablu, asigură o capacitate de transmisie de pînă la 1 *Gbit/s*. **Cablul optic** constă din fibre de sticlă sau din plastic transparent, acoperite cu un înveliș de protecție. Semnalul optic, emis de o sursă laser, se propagă prin fibră și este recepționat de o celulă fotosensibilă. Capacitatea de transmisie a unui cablu optic poate ajunge la valoarea de 1 *Tbit/s*.

Liniile cu microunde sînt formate din stații de retransmisie ce operează în banda de unde centimetrice. Pe Pămînt aceste stații se amplasează în raza vizibilității directe a antenelor, la o distanță de 40–50 de kilometri una de alta. În cazul liniilor cosmice stațiile respective se amplasează pe sateliți. Capacitatea de transmisie a liniilor cu microunde este de ordinul 10 *Gbit/s*.

În funcție de **aria de răspîndire** a calculatoarelor dintr-o rețea, există următoarele **tipuri de rețele**:

- rețele locale;
- rețele regionale;
- rețele globale.

În **rețelele locale** calculatoarele au o arie mică de răspîndire (pînă la 2 km) și deserveșc o singură instituție. Rețelele locale sînt formate, de regulă, din calculatoarele care se află în aceeași clădire sau într-un grup de clădiri. De obicei, ca linii de transmisie se utilizează cablurile cu fire torsadate și cablurile coaxiale.

Rețelele regionale acoperă aria unui oraș sau a unui sector. Liniile de comunicație se realizează prin cabluri coaxiale sau stații mici de transmisie/recepție, denumite **radiomodemuri**.

Rețelele globale acoperă suprafața unei țări, suprafața unui continent sau chiar suprafața mai multor continente. Ca linii de transmisie se utilizează cablurile optice și liniile cu microunde (terestre sau prin satelit).

Avantajul principal al rețelelor constă în **partajarea** sau, cu alte cuvine, **utilizarea în comun** a datelor, a programelor și a calculatoarelor din rețea.

De exemplu, în cazul unei rețele locale pot fi partajate fișierele, discurile de capacitate mare, imprimantele, cititoarele de desene și alte periferice. Evident, fiind accesibile pentru mai mulți utilizatori, echipamentele periferice respective vor fi utilizate mai eficient. Totodată, specialiștii instituției în cauză pot lucra în echipă asupra unor proiecte comune: bugetul anual, planul de vînzări, actualizarea bazelor de date etc.

În cazul rețelelor globale, colective de cercetători din diferite țări pot efectua calcule complexe pe un supercalculator unic în lume sau analiza în comun rezultatele unui experiment științific foarte costisitor. Pe baza rețelelor examinate sînt create diverse servicii: transferul de fișiere, poșta electronică, difuzarea noutăților, conversații pe grupuri de interese, jocuri electronice, publicitate, transferul banilor etc.

Întrebări și exerciții

- ❶ Numiți factorii care au contribuit la apariția rețelelor de calculatoare.
- ❷ Care sînt neajunsurile sistemelor centralizate de calcul?
- ❸ Numiți componentele principale ale unei rețele de calculatoare.
- ❹ Explicați destinația structurii de comunicație.
- ❺ Care sînt funcțiile adaptorului de rețea? Cum se identifică calculatoarele din componența unei rețele? Determinați tipul adaptorului de rețea cu care lucrați dvs.
- ❻ Din ce este formată o structură de comunicație?
- ❼ Care este destinația unui modem? Dar a unui radiomodem?
- ❽ Numiți capacitățile de transmisie a următoarelor linii de comunicație:
 - cablu cu fire torsadate;
 - cablu coaxial;
 - cablu optic;
 - linie cu microunde.
- ❾ Estimați durata de transmisie a unui film video (≈ 800 Gbiți) prin liniile de comunicație pe care le cunoașteți dvs.
- ❿ Determinați tipul liniilor de comunicație din structura rețelelor cu care lucrați dvs.
- ⓫ Cum se clasifică rețelele în funcție de aria de răspîndire?
- ⓬ Determinați tipul rețelei (locală, regională sau globală) cu care lucrați dvs.
- ⓭ Care sînt avantajele rețelelor de calculatoare? Ce servicii oferă o rețea de calculatoare?

7.2. Tehnologii de cooperare în rețea

Resursele unei rețele de calculatoare sînt echipamentele periferice, liniile de comunicație, calculatoarele propriu-zise, fișierele, bazele de date, programele executabile etc. Utilizarea eficientă a acestor resurse presupune lucrul în comun sau, cu alte cuvinte, cooperarea calculatoarelor și programelor ce rulează pe ele.

Numim tehnologie de cooperare modul cum este organizată funcționarea în comun a calculatoarelor și programelor din rețea.

Cel mai frecvent în rețelele de calculatoare se utilizează tehnologiile client-server și egal-la-egal.

În **tehnologia client-server** o resursă comună, de exemplu, imprimanta color sau discul de mare capacitate, este gestionată de un calculator dedicat, denumit **server**. Calculatorul care dorește să aibă acces la aceste resurse se numește **client**. Pentru a utiliza resursele respective, orice client trimite serverului o cerere. Serverul analizează cererile primite și, în funcție de statutul fiecărui client, le acceptă sau le respinge (fig. 7.3).

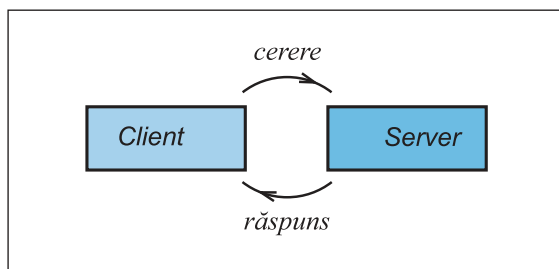


Fig. 7.3. Modelul client-server

Evident, calculatorul care gestionează fișiere comune se va numi **server de fișiere**, iar cel care gestionează imprimantele – **server de imprimare**.

Calculatorul care gestionează liniile de comunicație, alte resurse comune și, posibil, asigură accesul la rețele externe, se numește **server de rețea**. De regulă, anume pe acest calculator rulează și sistemul de operare al rețelei. Celelalte calculatoare din rețea au caracteristici mai modeste și se numesc **stații de lucru**.

Avantajele principale ale tehnologiei client-server sînt:

- utilizarea eficientă a echipamentelor scumpe;
- distribuirea rațională a lucrărilor între calculatoare în funcție de puterea lor;
- protecția sporită a datelor importante, ele fiind păstrate numai pe servere.

Cu regret, tehnologiile client-server sînt complicate și necesită calculatoare performante.

În **tehnologia egal-la-egal** funcțiile tuturor calculatoarelor din rețea sînt identice. Fiecare calculator din rețea funcționează atît ca server, cît și ca stație de lucru, oferind pentru uzul public resursele de care dispune, în mod curent, unele fișiere de pe discul fix, unitatea de disc optic, imprimanta etc. Fiind simplă, tehnologia dată se folosește în rețelele locale mici. În cazul rețelelor mari, tehnologia egal-la-egal nu permite protecția sigură a datelor.

În mod similar, tehnologiile client-server se aplică și pentru organizarea lucrului în comun a două sau a mai multe programe.

Programul în curs de execuție care oferă servicii se numește program server, iar programele care apelează la aceste servicii se numesc programe client.

De exemplu, un program server ce gestionează o bază de date execută următoarele funcții:

- asigură protecția și securitatea datelor;
- recepționează și, dacă clientul are autorizațiile respective, execută cererile de modificare a datelor;
- recepționează cererile de citire a datelor și, în funcție de statutul clientului, permite sau interzice accesul la datele respective;
- completează un registru în care înscrie toate operațiile efectuate în baza de date.

Programul client asigură interacțiunea utilizatorului cu baza de date și realizează următoarele funcții:

- oferă utilizatorului o interfață simplă și comodă;
- verifică și editează datele introduse de utilizator;
- adresează cereri programului server;
- afișează informațiile primite din baza de date.

Programele server și programele client pot rula pe un singur calculator sau pe calculatoare diferite. În ultimul caz, prelucrarea de date este **distribuită**. Transferul de date între programul client și programul server se realizează prin structura de comunicație (fig. 7.2). Calculatorul pe care rulează un program server se numește **gazdă** (în engleză *host*).

De obicei, în rețelele locale programul client se execută pe stațiile de lucru, iar programele server rulează pe serverul de rețea. În cazul rețelelor regionale sau globale pe fiecare calculator performant rulează mai multe programe server ce oferă diverse servicii programelor client de pe alte calculatoare.

Întrebări și exerciții

- ❶ Explicați termenul *tehnologii de cooperare în rețea*. Ce tehnologii de cooperare în rețea cunoașteți?
- ❷ Cum este organizată funcționarea calculatoarelor din rețea în cazul tehnologiei client-server?
- ❸ Care sînt avantajele și dezavantajele tehnologiei client-server?
- ❹ Explicați destinația serverelor de rețea, a serverului de fișiere, a serverului de imprimare și a stației de lucru.
- ❺ Cum este organizată funcționarea calculatoarelor din rețea în cazul tehnologiei egal-la-egal? Care sînt avantajele și dezavantajele acestei tehnologii?
- ❻ Determinați tehnologiile de cooperare realizate în rețeaua cu care lucrați dvs. Există oare în rețea un server de fișiere și/sau un server de imprimare?
- ❼ Există în rețeaua cu care lucrați dvs. un server de rețea? Argumentați răspunsul.
- ❽ Este oare realizată în rețeaua cu care lucrați dvs. tehnologia egal-la-egal? Argumentați răspunsul.

- 9 Cum interacționează programele în curs de execuție în cazul tehnologiei client-server?
- 10 Numiți funcțiile unui program server și ale unui program client.
- 11 Explicați termenul *calculator-gazdă*.
- 12 O companie de transporturi aeriene a deschis agenții de vânzare a biletelor în mai multe orașe din țară. Datele despre toate rutele aeriene și locurile disponibile se păstrează în calculatorul din sediul central al companiei. Care tehnologii de rețea ar asigura lucrul eficient al agențiilor de vânzare a biletelor? Sînt oare necesare programe server și programe client? Ce funcții ar realiza aceste programe?
- 13 Jocurile electronice colective presupun existența a 5-10 calculatoare reunite în rețea. În astfel de jocuri fiecare luptă contra tuturor. Calculatoarele respective oferă pentru uzul public o partiție de pe discul fix. Ce tehnologie de rețea ar asigura lucrul în comun al calculatoarelor?
- 14 Elaborați o tehnologie de cooperare în rețea pentru calculatoarele din:
 - a) depozitele unei firme;
 - b) sălile de expoziție ale unui muzeu;
 - c) sălile de lectură ale unei biblioteci;
 - d) casele unui magazin care acceptă carduri bancare;
 - e) sistemul de eliberare a banilor lichizi prin intermediul automatelor bancare;
 - f) sistemul de verificare operativă a numerelor de înmatriculare a automobilelor (fiecare echipaj de poliție este dotat cu un microcalculator și un radiomodem);
 - g) laboratorul de informatică.

7.3. Topologia și arhitectura rețelelor

Structura de comunicație a unei rețele (*fig. 7.2*) asigură transferul de date între calculatoare. De obicei, datele de transmis sînt grupate în pachete.

Un **pachet de date** conține următoarele informații:

- adresa destinatarului;
- datele propriu-zise;
- informații de control;
- adresa expeditorului.

Se observă că pachetul de date poate fi tratat ca un plic obișnuit care circulă într-un sistem tradițional de poștă. Drumul parcurs de un pachet de date depinde de topologia rețelei.

Numim topologie a rețelei configurația geometrică a legăturilor între calculatoare.

Topologiile concrete ale rețelelor actuale sînt formate prin utilizarea structurilor de bază: stea, inel, magistrală, distribuită etc. (*fig. 7.4*).

În cazul **topologiei stea** legătura între două calculatoare C_i , C_j ale rețelei are loc prin intermediul calculatorului central C_1 . Din acest motiv, calculatorul C_1 , denumit **calculator principal**, are un rol important în funcționarea rețelei, efectuînd dispecerizarea pachetelor de date. Evident, defectarea calculatorului prin-

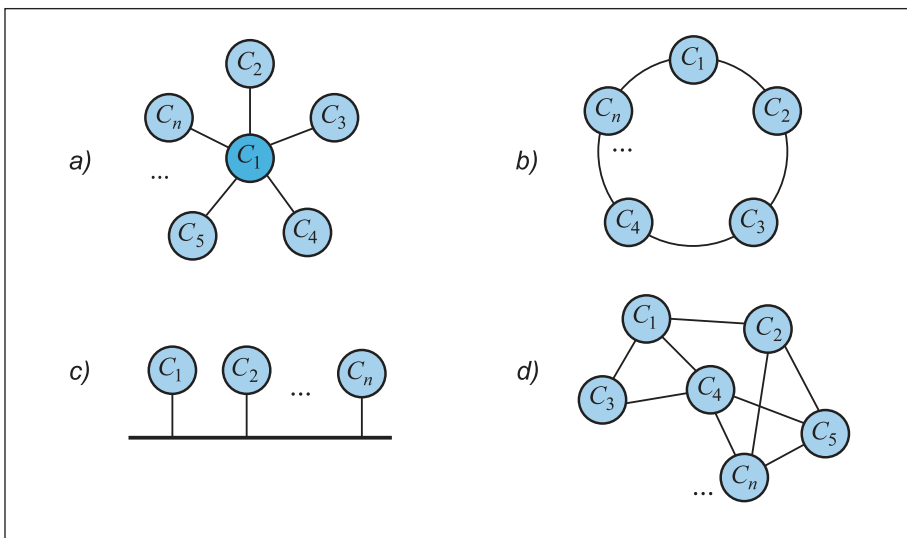


Fig. 7.4. Topologii de rețea:
a – stea; b – inel; c – magistrală; d – distribuită

cipal întrerupe funcționarea întregii rețele. Prin urmare, calculatorul principal trebuie să fie foarte fiabil.

În **topologia inel** legăturile între calculatoare formează o buclă închisă. Pachetul expedit de calculatorul C_i este transmis calculatorului C_{i+1} , care la rîndul său îl transmite calculatorului C_{i+2} etc. pînă cînd pachetul ajunge la calculatorul destinat C_j . Întrucît defectarea oricărui calculator întrerupe funcționarea întregii rețele, toate calculatoarele $C_1, C_2, \dots, C_n$ trebuie să fie foarte fiabile.

În **topologia magistrală** există un singur canal de comunicație la care sînt conectate toate calculatoarele. Fiecare calculator „spionează” magistrala și interceptează pachetele adresate lui. Orice calculator C_i poate expedia un pachet numai atunci cînd magistrala este liberă.

Rețelele bazate pe topologia de tip magistrală sînt foarte fiabile, întrucît comunicarea între calculatoarele C_i, C_j va avea loc chiar și în cazul în care toate celelalte calculatoare nu funcționează.

În **topologiile distribuite** între fiecare pereche de calculatoare există mai multe căi de transmisie a datelor. De exemplu, un pachet de date expedit de calculatorul C_1 calculatorului C_n (fig. 7.4d) poate ajunge la destinație pe drumul $C_1 - C_2 - C_5 - C_n$, iar altul pe drumul $C_1 - C_4 - C_n$. Evident, rețeaua rămîne funcțională chiar dacă unul sau mai multe calculatoare și linii de comunicație nu funcționează.

De regulă, topologiile de tip stea, inel sau magistrală se utilizează în cazul rețelelor locale. Rețelele regionale și cele globale au o topologie distribuită. Integrarea rețelelor locale în rețele regionale și rețele globale se face conform topologiilor de bază din figura 7.4, fiecare nod $C_1, C_2, \dots, C_n$ reprezentînd o subrețea. Prin urmare, ajungem la o structură ierarhică cu o diversitate de legături.

Setul de reguli pentru gestionarea schimbului de date într-o rețea este numit protocol de comunicație sau, pur și simplu, protocol.

Un **protocol** definește modul de adresare a calculatoarelor, lungimea și componența pachetelor de date, algoritmul de depistare și corectare a erorilor, modul de conectare fizică a adaptoarelor și cablurilor de rețea etc. La apariția primelor rețele de calculatoare fiecare producător de echipamente de calcul avea propriile sale protocoale de comunicație, ceea ce făcea imposibilă interconectarea calculatoarelor de proveniențe diferite. Acest neajuns a fost înlăturat prin standardizarea protocoalelor. Amintim că **standardul** reprezintă un document în care se reglementează calitatea, caracteristicile, forma etc. ale unui produs. Standardele internaționale sînt elaborate de către Organizația Internațională pentru Standardizare (*ISO – International Standards Organisation*). Un alt organism internațional care joacă un rol important în standardizarea în domeniile electronicii și tehnicii de calcul este Institutul Inginerilor Electricieni și Electroniști (*IEEE – Institute of Electrical and Electronics Engineers*).

Numim arhitectură a rețelei modul în care este concepută rețeaua: topologia, protocoalele de comunicație, tehnologiile de cooperare în rețea.

În continuare prezentăm cele mai răspîndite arhitecturi de rețele.

Ethernet (rețea în eter) – rețele locale realizate în conformitate cu standardul *IEEE 802.3*. Utilizează o magistrală din cablu cu fire torsadate, cablu coaxial sau cablu cu fibre optice. Viteza de transmisie a datelor ajunge la 100 *Mbiți/s*. Arhitectura a fost elaborată de firmele *XEROX*, *Intel* și *DEC*.

Token-Ring (inel cu jeton) – rețele locale realizate în conformitate cu standardul *IEEE 802.5*. Utilizează un inel din cablu cu fire torsadate sau cablu coaxial. Viteza de transmisie a datelor este de 16 *Mbiți/s*. Arhitectura a fost elaborată de firma *IBM*.

DATAKIT – rețele locale, regionale sau globale elaborate de firma *Bell Laboratories*. Sub aspect topologic constă dintr-o mulțime de stele interconectate. În cazul cablului cu fibre optice se atinge viteza de transmisie de 1,5 *Gbiți/s*.

SNA (Sistem Network Architecture) – o arhitectură elaborată de firma *IBM* pentru rețelele locale, regionale și globale. Protocoalele acestei arhitecturi au stat la baza standardelor *ISO*. Topologia inițială era de tip stea, în prezent este o topologie distribuită, suportînd și rețele locale.

ARPANET – o arhitectură concepută de mai multe universități și corporații sub egida Ministerului Apărării al SUA (*Advanced Research Projects Agency*). Arhitectura dată se bazează pe o topologie distribuită și folosește diferite linii de comunicație, de la liniile telefonice pînă la liniile cu microunde prin satelit. Liniile respective conectează supercalculatoare separate și diverse rețele locale sau regionale, răspîndite pe aproape jumătate din suprafața terestră. Arhitectura *ARPANET* include următoarele protocoale:

- protocolul **IP** (*Internet Protocol*) destinat interconectării rețelelor locale, regionale și globale;
- protocolul serviciilor bazate pe conexiuni, numit **TCP** (*Transmission Control Protocol*);
- un protocol de transfer de fișiere, numit **FTP** (*File Transfer Protocol*);

- un protocol pentru poșta electronică, numit **SMTP** (*Simple Mail Transfer Protocol*);
 - un protocol de atașare de la distanță a calculatoarelor, numit **TELNET**.
- Pe baza arhitecturii **ARPANET** a fost concepută rețeaua globală de calculatoare **Internet**.

Întrebări și exerciții

- 1 Explicați termenul *pachet de date*. Ce informații conține un pachet de date?
- 2 Explicați termenul *topologia rețelei*.
- 3 Care sînt topologiile de bază ale rețelelor? Numiți avantajele și dezavantajele fiecărei topologii de bază.
- 4 Explicați cum se transmit pachetele de date în următoarele topologii de bază: stea, inel, magistrală, distribuită.
- 5 Precizați drumul pe care îl va parcurge un pachet trimis de calculatorul C_2 calculatorului C_4 (fig. 7.4a).
- 6 Găsiți drumul pe care îl parcurg pachetele expediate de calculatorul C_2 calculatorului C_4 (fig. 7.4b).
- 7 Cîte căi de transmitere a pachetelor există între calculatoarele C_1 , C_n din figura 7.4d? Care este calea cea mai scurtă?
- 8 Ce se va întîmpla cu rețelele din figura 7.4 dacă iese din funcție calculatorul C_1 ? Dar dacă iese din funcție calculatorul C_2 ?
- 9 Determinați topologia rețelei locale cu care lucrați dvs. Enumerați avantajele și dezavantajele acestei topologii.
- 10 Care este destinația unui protocol? Ce norme conține un protocol?
- 11 Argumentați necesitatea standardizării protocoalelor. Cine elaborează standardele respective?
- 12 Aflați protocoalele utilizate de rețeaua locală cu care lucrați dvs.
- 13 Explicați modul cum se integrează rețelele locale în rețele regionale și rețele globale. Desenați topologia rețelelor regionale sau globale la care aveți acces dvs.
- 14 Explicați termenul *arhitectura rețelei*.
- 15 Dați exemple de arhitecturi ale rețelelor locale, regionale și globale.
- 16 Caracterizați arhitectura rețelei cu care lucrați dvs.
- 17 Enumerați protocoalele utilizate în arhitectura **ARPANET**. Sînt oare aplicate aceste protocoale în rețeaua cu care lucrați dvs.?

7.4. Rețeaua Internet

Rețeaua globală **Internet** se bazează pe o topologie distribuită și include calculatoare separate, subrețele locale, regionale sau globale (fig. 7.5).

Interconectarea rețelelor se realizează cu ajutorul unor echipamente dedicate de rețea, denumite porți sau rutere. **Poarta** (*gateway*) este un calculator specializat destinat interconectării a două rețele ce utilizează protocoale total diferite de co-

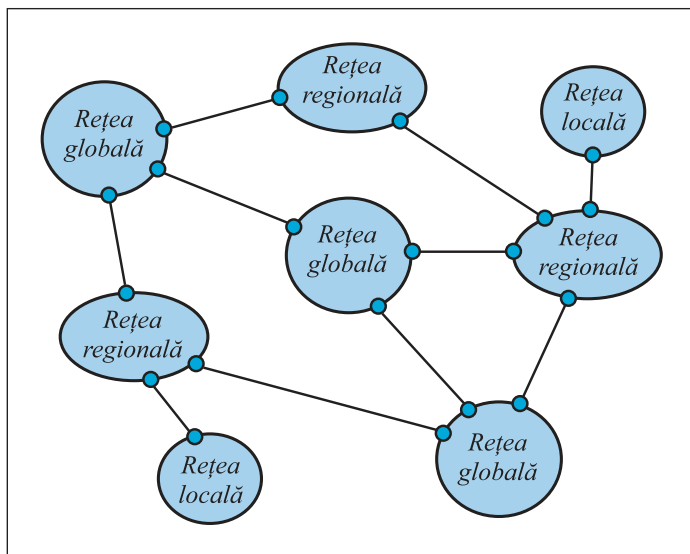


Fig. 7.5. Topologia Internetului

municație. **Ruterul** (*router*) este un calculator dedicat care interconectează rețelele ce folosesc protocoale identice și sînt utilizate pentru a determina cea mai bună cale de transmitere a pachetelor. Calculatoarele conectate la *Internet* se numesc **gazde** (*host*). Funcționarea rețelei este reglementată de circa 100 de protocoale.

Identificarea calculatoarelor în cadrul rețelei se face cu ajutorul **adreselor Internet**. Acestea pot fi de două tipuri: adrese numerice și adrese simbolice.

O **adresă numerică** este formată din 32 de cifre binare (4 octeți) și are structura prezentată în figura 7.6.

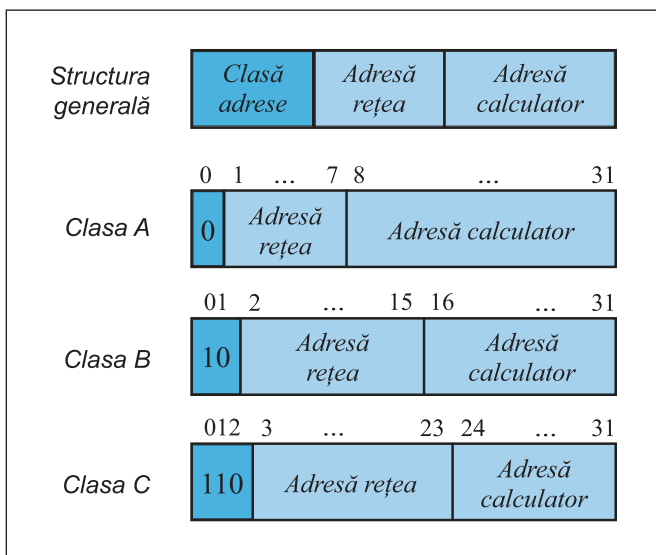


Fig. 7.6. Structura adreselor numerice

Întrucît *Internetul* este o „rețea de rețele”, adresa numerică conține adresa subrețelei (cîmpul *Adresă Rețea*) și adresa calculatorului în cadrul subrețelei (cîmpul *Adresă Calculator*). În funcție de numărul maximal de calculatoare pe care le poate identifica în cadrul unei subrețele, adresele se împart în clasele *A*, *B* și *C*. Adresele din clasele *D* și *E* au o destinație specială. Caracteristica adreselor *Internet* este prezentată în *tabelul 7.1*.

Tabelul 7.1

Caracteristica adreselor numerice

Clasa	Numărul de adrese disponibile	
	de rețea	de calculatoare
<i>A</i>	$2^7=128$	$2^{24} = 16\ 777\ 216$
<i>B</i>	$2^{14}=16\ 384$	$2^{16} = 65\ 536$
<i>C</i>	$2^{21}=2\ 097\ 152$	$2^8 = 256$

Adresele din clasa *A* sînt distribuite rețelelor mari, în special rețelelor globale. O astfel de rețea poate include circa 16 milioane de calculatoare. Adresele din clasa *B* sînt distribuite rețelelor medii, în special rețelelor regionale. O astfel de rețea poate include circa 65 mii de calculatoare. Adresele din clasa *C* sînt rezervate rețelelor relativ mici, care includ pînă la 256 de calculatoare. Fiecare adresă *Internet* este unică. Adresele sînt atribuite calculatoarelor de **Centrul Informațional al Rețelei** (*Network Information Center*).

De exemplu, numărul binar

10010010 00110011 00001001 11110111

este o adresă de clasa *B* și identifică calculatorul numărul

00001001 11110111

din cadrul rețelei

010010 00110011.

Pentru comoditate, adresele binare se exprimă în formă zecimală, transformînd separat fiecare octet. Numerele zecimale corespunzătoare fiecărui octet se delimitează prin puncte.

În cazul exemplului de mai sus obținem:

$$(10010010)_2=(146)_{10};$$

$$(00110011)_2=(51)_{10};$$

$$(00001001)_2=(9)_{10};$$

$$(11110111)_2=(247)_{10}.$$

Prin urmare, în forma zecimală adresa respectivă va fi 146.51.9.247.

Adresele în formă zecimală și cu atît mai mult cele în formă binară sînt incommode pentru publicul larg. Din acest motiv, mai frecvent sînt utilizate adresele simbolice.

O **adresă simbolică** este formată din numele calculatorului-gazdă și nume de domenii separate prin puncte. **Domeniul** reprezintă un grup de calculatoare organizate tematic sau geografic. Orice domeniu poate fi împărțit în subdomenii, ajungându-se astfel la o structură ierarhică. Numele de domenii se indică în ordinea creșterii ariei de cuprindere.

De exemplu, adresele simbolice

c1.lme.ch.md

c5.lme.ch.md

specifică calculatoarele c1 și c5 din domeniul lme (Liceul „Mihai Eminescu”).

Adresele simbolice

c1.lic.ch.md

c9.lic.ch.md

specifică calculatoarele c1 și c9 din domeniul lic (Liceul „Ion Creangă”). Domeniile lme și lic sînt subdomenii ale domeniului ch (Chișinău). La rîndul său, ch este un subdomeniu al domeniului md (Republica Moldova).

În mod similar, adresele simbolice

rector.ase.men.ro

decan.ase.men.ro

specifică calculatoarele rector și decan din domeniul ase (Academia de Studii Economice). Domeniul ase este un subdomeniu al domeniului men (Ministerul Educației Naționale), iar men este un subdomeniu al domeniului ro (România).

În mod normal, domeniul de cel mai înalt nivel este țara (md, ro, us etc.) sau tipul instituției (com – comercială, mil – militară, edu – de educație etc.).

Relațiile de incluziune între domenii pot fi reprezentate cu ajutorul diagramei *Euler*, frecvent utilizate în teoria mulțimilor. Pentru exemplificare, în *figura 7.7* sînt prezentate astfel de diagrame pentru unele adrese simbolice din domeniul md.

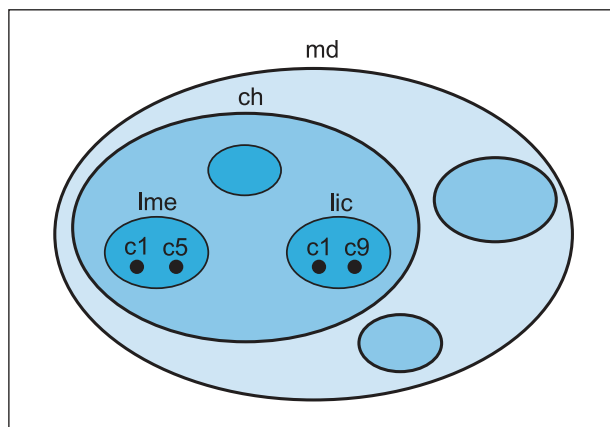


Fig. 7.7. Diagrame Euler pentru adrese simbolice

Pentru a **transforma** adresele simbolice în adrese numerice și invers, în fiecare domeniu există un **server de nume** (*name server*). Acest program gestionează domeniul respectiv fără intervenția serverelor ierarhic superioare. Prin urmare, în *Internet* nu există un calculator central care ar fi unicul responsabil de cele aproximativ 10^{10} adrese de rețea.

În cazul exemplului de mai sus (*fig. 7.7*), adresele calculatoarelor c1 și c5 sînt procesate în serverul lme, iar adresele calculatoarelor c1 și c9 – de serverul lic. Serverul ch va lucra numai cu adresele serverelor lic și lme, fără să se implice în procesarea adreselor de calculatoare din aceste domenii. În mod similar, serverul men nu procesează adresele calculatoarelor rector și decan, lăsînd acest lucru în obligația serverului ase.

Întrebări și exerciții

- ❶ Desenați topologia rețelei *Internet*. Cum se interconectează subrețelele în cadrul *Internetului*?
- ❷ Care este destinația unei porți? Dar a unui ruter?
- ❸ Cum se identifică calculatoarele în *Internet*? Care sînt avantajele și dezavantajele adreselor numerice? Dar ale adreselor simbolice?
- ❹ Care sînt clasele de adrese numerice? Pentru ce este necesară o astfel de clasificare?
- ❺ Explicați destinația cîmpurilor *Adresă rețea* și *Adresă calculator* ale unei adrese numerice.
- ❻ Cine atribuie adrese calculatoarelor din *Internet*?
- ❼ Determinați clasele următoarelor adrese. Precizați adresa subrețelei și adresa calculatorului în subrețea.

a) 45.201.19.63;

d) 192.109.58.170;

b) 201.165.213.91;

e) 15.21.207.250;

c) 154.36.79.200;

f) 217.15.69.113.

- ❽ Care sînt criteriile de grupare a calculatoarelor în domenii? Numiți domeniile de cel mai înalt nivel.
- ❾ Sînt date următoarele adrese simbolice:

a) c1.lme.ch.md;

f) c4.lme.ch.md;

b) c3.lme.ch.md;

g) c5.lme.ch.md;

c) c1.lic.ch.md;

h) c9.lic.ch.md;

d) director.lic.ch.md;

i) prof.lic.ch.md;

e) elev1.lic.ch.md;

j) elev4.lic.ch.md.

Precizați domeniile de calculatoare și relațiile de incluziune între domenii. Desenați diagramele *Euler* pentru adresele în cauză.

⑩ Desenați o diagramă *Euler* pentru adresele simbolice ce urmează. Precizați domeniile de calculatoare și relațiile respective de incluziune.

a) rector.ase.men.ro;

d) rector.ase.met.md;

b) decan.ase.men.ro;

e) decan.ase.met.md;

c) student.info.ase.men.ro;

f) student.cib.met.md.

⑪ Care este destinația unui server de nume? Ce adrese sînt procesate de un astfel de server?

⑫ Precizați serverele de nume pentru adresele din exercițiile 9 și 10. Indicați adresele pe care le procesează fiecare server.

⑬ Aflați adresa numerică și adresa simbolică a calculatorului cu care lucrați dvs. Precizați clasa de adresă, adresa subrețelei și adresa calculatorului în cadrul subrețelei. Desenați o diagramă *Euler* ce reprezintă relațiile de incluziune între domeniile la care aparține calculatorul dvs.

7.5. Servicii Internet

Rețeaua *Internet* oferă următoarea gamă de servicii:

- accesul calculatoarelor la distanță;
- transferul de fișiere;
- poșta electronică;
- știri și discuții;
- prezentarea și căutarea informațiilor etc.

Cooperarea calculatoarelor și programelor care oferă aceste servicii se bazează pe modelul client-server. De obicei, pe calculatorul beneficiarului de serviciu rulează programul client, iar pe calculatorul furnizorului de servicii rulează programul server.

Serviciul Telnet permite utilizatorului să aibă accesul la calculatoarele aflate la distanță. După stabilirea conexiunii, calculatorul utilizatorului devine un simplu terminal al calculatorului aflat la distanță. În continuare, utilizatorul poate lansa în execuție pe calculatorul respectiv diverse programe, poate vizualiza fișiere, schimba directoare etc. Protecția calculatoarelor și a datelor respective se asigură prin utilizarea parolelor. Serviciul Telnet se utilizează pentru folosirea în comun a unor resurse foarte scumpe, de exemplu, a supercalculatoarelor.

Serviciul transfer de fișiere sau, mai scurt, **serviciul FTP** (*File Transfer Protocol*) permite utilizatorului să copie fișiere de pe calculatoare situate în diverse puncte geografice. Acest serviciu oferă două moduri de transfer al fișierelor:

- modul binar, în care se păstrează secvența de biți a fișierului, astfel încît originalul și copia sînt identice bit cu bit;
- modul text, în care se transferă seturi de caractere în codul *ASCII*.

În general, pentru a avea acces la serverul FTP, clientul trebuie să introducă o parolă. Totuși există servere publice (*FTP anonymous*) care permit accesul la fișiere fără a fi nevoie de o parolă specială.

Serviciul de poștă electronică (*electronic mail* sau, prescurtat, *e-mail*) a copiat modul de funcționare a poștei obișnuite.

Scrisoarea electronică, denumită **mesaj** (*message*) include:

- adresa destinatarului;
- subiectul, exprimat în câteva cuvinte;
- adresa expeditorului;
- textul scrisorii;
- fișiere atașate opțional.

Fișierele atașate pot fi de orice natură: texte, imagini, programe etc.

Scrisorile sînt depuse în fișiere speciale, denumite **cutii poștale** (*mail box*). Adresa unei cutii poștale are forma:

`<nume cutie>@<Adresă calculator>`,

unde:

`<nume cutie>` este denumirea cutiei poștale, de obicei acesta este numele de familie al utilizatorului sau o abreviere;

`@` – simbolul „at” (la);

`<Adresă calculator>` – adresa simbolică a calculatorului client pe care este creată cutia poștală.

Exemple:

1) `petrescu@c1.lme.ch.md`

2) `florea@director.lic.ch.md`

3) `ionescu@c1.lme.ch.md`

4) `barbu@director.lic.ch.md`

Cititorii pot expedia scrisori autorilor acestui manual la adresa:

`Anatol_Gremalschi@yahoo.com`

Mesajele sînt transmise prin rețea de serverele de poștă care au rolul oficiilor și centrelor poștale tradiționale.

Serviciul de poștă electronică este foarte popular datorită avantajelor sale incontestabile, și anume: viteză, posibilitatea de a atașa la scrisori fișiere de orice natură, facilități avansate de redactare.

Cel mai modern serviciu de prezentare și căutare a informațiilor în *Internet* este **serviciul WWW** (*World Wide Web* – Pinza Mondială de Păianjen). În acest serviciu informația este prezentată în formă de pagini Web.

Pagina Web este un fișier scris în limbajul HTML (*Hypertext Markup Language* – Limbaj pentru marcarea hipertextului) și poate conține, în afară de informații propriu-zise, referințe la alte pagini Web. Paginile referite se pot afla pe același calculator sau pe calculatoare situate în diverse puncte geografice (*fig. 7.8*).

Tehnologia de cooperare în cadrul serviciului WWW este de tipul client-server. Utilizatorul care dorește să ofere publicului larg anumite informații instalează pe

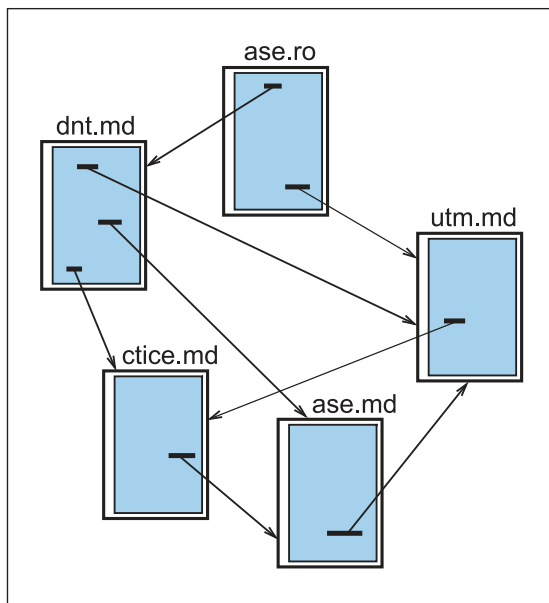


Fig. 7.8. Pagini Web

calculatorul său un program server și elaborează una sau mai multe pagini Web. Serverul interceptează cererile sosite de la alte calculatoare și asigură accesul la paginile respective. Calculatorul pe care sînt instalate paginile Web și serverul WWW se numește **site** (*site* – sediu, reședință).

Programul client asigură transferul și afișarea pe ecran a paginilor Web citite de pe diverse calculatoare din *Internet*. Imediat după pornire, el afișează propria pagină de referință (*home page* – pagina de acasă) și așteaptă indicațiile utilizatorului. Cînd utilizatorul activează o referință, programul client stabilește o conexiune cu serverul Web și copiază de la el pagina specificată în referință. Pagina copiată este afișată pe ecran.

În continuare, utilizatorul activează o altă referință, programul client stabilește din nou conexiunea cu un calculator din rețea, din nou se va citi o pagină Web etc. Cu alte cuvinte, utilizatorul „răsfoiește” paginile Web de pe diverse calculatoare, indiferent de poziția lor geografică. Din acest motiv programele client sînt numite **programe de răsfoire** sau **programe de explorare** (în engleză *browser* sau *explorer*).

În cadrul serviciului WWW resursele rețelei se specifică cu ajutorul unor adrese speciale, denumite adrese URL (*Uniform Resource Locator* – Locator Uniform de Resurse). Aceste adrese au forma:

```
<protocol>: // <Adresă simbolică>[:<port>]/<cale>/<fișier>
```

unde:

<protocol> – specifică denumirea protocolului pentru transferul datelor prin rețea;

<Adresă simbolică> – adresa calculatorului ce conține fișierul respectiv;

<port> – portul de acces (opțional);
<cale>/<fișier> – specificarea fișierului.

Pentru exemplificare, prezentăm câteva adrese URL ce conțin informații interesante:

<http://www.edu.md> – site-ul Ministerului Educației și Tineretului, Moldova;

<http://www.ctice.edu.md> – site-ul Centrului de Tehnologii Informaționale și Comunicaționale în Educație, Moldova;

<http://www.dnt.md> – site-ul Asociației *Dynamic Network Tehnologies*, Moldova;

<http://www.itc.ro/museum/museum.html> – site-ul Muzeului Național de Artă, România;

<http://www.nmsi.ac.uk> – site-ul Muzeului de Știință și Industrie, Marea Britanie;

<http://www.nasa.gov> – site-ul agenției NASA, Statele Unite ale Americii.

Amintim că notația *http* specifică protocolul de transfer al hypertextelor (*Hypertext Transfer Protocol*).

În prezent, numărul fișierelor din rețeaua *Internet* este de ordinul miliardelor. Evident, nici nu poate fi vorba de căutarea informației necesare prin citirea individuală a fiecărui fișier. Pentru a simplifica căutarea informației, în cadrul rețelei *Internet* au fost create servere de căutare (*search engine*).

Serverul de căutare este un calculator puternic care explorează încontinuu rețeaua și citește paginile Web sau alte informații prezentate publicului larg. Acestea sînt clasificate în funcție de datele pe care le conțin, iar adresele lor sînt reținute în baza de date de pe server.

Programul client adresează serverului de căutare o cerere în care indică de ce fel de informații are nevoie. Serverul interoghează baza de date și transmite clientului o listă de adrese la care pot fi găsite informațiile cerute.

Pentru exemplificare amintim serverele de căutare frecvent utilizate:

<http://www.yahoo.com> – serverul YAHOO (*Yet Another Hierarhicaly Organized Oracle* – încă un oracol organizat ierarhic) al companiei *Yahoo! Inc.*;

<http://www.google.com> – serverul GOOGLE al corporației *Google Inc.*;

<http://www.bing.com> – serverul BING al corporației *Microsoft Inc.*;

<http://www.yandex.ru> – serverul YANDEX al companiei „Яндекс”;

<http://www.infoseek.com> – serverul Infoseek al firmei *Infoseek Corp.*

Accesul la aceste servere este gratuit.

Întrebări și exerciții

- 1 Care este gama de servicii oferite de *Internet*? Cum colaborează calculatoarele din rețea în procesul prestării unui serviciu?
- 2 Care este destinația serviciului *FTP*? Ce fișiere pot fi transferate prin acest serviciu?
- 3 Studiați programul *FTP* instalat pe calculatorul dvs. Copiați câteva fișiere de pe serverele *FTP anonymous* sau de pe alte servere la care aveți acces.
- 4 De ce depinde viteza de transfer al fișierelor? Determinați viteza de transfer a câtorva fișiere de pe serverele *FTP* aflate în Republica Moldova, România și Statele Unite ale Americii.

- ⑤ Cum se specifică adresele în cadrul serviciilor de poștă electronică?
- ⑥ Explicați cum interacționează calculatoarele client și serverele de poștă în cadrul serviciului respectiv.
- ⑦ Ce informații conține o scrisoare electronică? Care dintre ele sînt opționale?
- ⑧ Explicați cum se formează o adresă poștală. Aflați adresele poștale ale prietenilor dvs.
- ⑨ Studiați programul de poștă electronică instalat pe calculatorul la care lucrați dvs. Verificați dacă acest program oferă următoarele facilități:
 - utilizarea mai multor cutii poștale;
 - clasificarea și păstrarea scrisorilor în dosare;
 - elaborarea scrisorilor conform unor șabloane;
 - cifrarea și descifrarea corespondenței;
 - semnalarea momentelor cînd a sosit corespondența;
 - verificarea faptului că scrisorile expediate au ajuns la destinație;
 - utilizarea semnăturilor electronice.
- ⑩ Care sînt avantajele poștei electronice? Poate înlocui acest serviciu poșta tradițională?
- ⑪ Formați grupe de cîte patru utilizatori ai poștei electronice și determinați experimental viteza cu care se transmite corespondența.
- ⑫ Ce informații conține pagina Web? Cum formează aceste pagini o „pînză de păianjen”?
- ⑬ Care este destinația unui server Web? Dar a unui client Web? Cum interacționează aceste programe?
- ⑭ Explicați modul de funcționare a unui client Web. Cum găsește acest program paginile Web amplasate pe diferite calculatoare?
- ⑮ Explicați modul de specificare a resurselor în *Internet* cu ajutorul adreselor URL. Care este semnificația cîmpurilor acestor adrese?
- ⑯ Studiați programul de explorare a *Internetului* instalat pe calculatorul cu care lucrați dvs. Ce facilități oferă acest program? Citiți paginile Web adresele cărora sînt indicate în acest paragraf.
- ⑰ Care este destinația unui server de căutare? Ce servicii oferă un astfel de server?
- ⑱ Găsiți cu ajutorul unui server de căutare furnizorii de servicii *Internet* din Republica Moldova.
- ⑲ În afară de serviciile studiate în acest paragraf, rețeaua *Internet* oferă și alte servicii, cum ar fi: *Archie*, *Gopher*, *WAIS*, buletine de știri, discuții etc. Folosind un server de căutare, aflați informația respectivă despre aceste servicii.
- ⑳ Elaborați paginile Web ale clasei și ale liceului dvs.

Test de autoevaluare nr. 7

1. Estimați durata de transmisie a unui film video în formă arhivată (≈ 750 Mbiți) prin următoarele linii de comunicație:
 - a) linie telefonică de 36 Kbiți/s;
 - b) cablu cu fire torsadate de 1 Mbit/s;
 - c) cablu coaxial de 1 Gbit/s;
 - d) cablu optic de 1 Tbit/s.
2. Care dintre afirmațiile ce urmează sînt adevărate:
 - a) rețelele locale acoperă aria unei localități;
 - b) rețelele globale acoperă suprafața unei țări, suprafața unui continent sau chiar suprafața mai multor continente;
 - c) în general, rețelele regionale deserveșc localitățile unei singure companii;
 - d) de obicei, rețelele locale deserveșc o singură instituție?
3. Numiți componentele principale ale unei rețele de calculatoare.
4. Care dintre afirmațiile ce urmează sînt adevărate:
 - a) în tehnologia egal-la-egal funcțiile tuturor calculatoarelor din rețea sînt identice;
 - b) serverul unei rețele are aceleași funcții ca și oricare alt calculator din rețea;
 - c) în tehnologia client-server o resursă comună este gestionată de un calculator dedicat;
 - d) orice calculator-client poate accesa datele de pe oricare alt calculator din rețea?
- 5*. Care dintre afirmațiile ce urmează sînt adevărate:
 - a) în topologia stea legătura între două calculatoare ale rețelei are loc prin intermediul calculatorului central;
 - b) în topologiile distribuite legătura între două calculatoare ale rețelei are loc prin intermediul magistralei;
 - c) în topologia magistrală legătura între două calculatoare ale rețelei are loc prin intermediul calculatorului central;
 - d) în topologia inel toate calculatoarele au funcții identice?
- 6*. Care este destinația unui protocol de rețea?
- 7*. Explicați semnificația termenului *arhitectura rețelei*.
8. Desenați topologia rețelei *Internet*.
9. Care dintre afirmațiile ce urmează este adevărată:
 - a) *Internetul* este o rețea locală ce are acces la alte rețele locale, regionale sau globale;
 - b) *Internetul* reprezintă totalitatea rețelelor dintr-o țară;
 - c) *Internetul* este o rețea a marilor companii internaționale;
 - d) *Internetul* reprezintă o rețea distribuită, formată din rețele globale, regionale și locale interconectate între ele;
 - e) *Internetul* este format din toate calculatoarele lumii, interconectate între ele?
10. Cum se identifică calculatoarele în *Internet*?
- 11*. Determinați clasa adresei numerice *Internet* 214.121.216.109. Precizați adresa subrețelei și adresa calculatorului în subrețea.

* Numai pentru profilul real.

12. Desenați o diagramă *Euler* pentru adresele simbolice ce urmează. Precizați domeniile de calculatoare și relațiile respective de incluziune.

- | | |
|--|--|
| a) <code>directie.orhei.md</code> ; | d) <code>directie.cahul.md</code> ; |
| b) <code>contabilitate.orhei.md</code> ; | e) <code>contabilitate.cahul.md</code> ; |
| c) <code>presedinte.orhei.md</code> ; | f) <code>presedinte.calarasi.md</code> . |

13. Indicați corespondența între denumirile serviciilor *Internet* (coloana din stînga) și destinația acestora (coloana din dreapta):

- | | |
|-------------|---|
| (1) Telnet; | (a) transferul fișierelor de pe calculatorul aflat la distanță pe calculatorul utilizatorului; |
| (2) FTP; | (b) căutarea informațiilor pe discul magnetic al calculatorului și afișarea lor la ecran; |
| (3) E-mail; | (c) editarea imaginilor grafice și transmiterea lor în <i>Internet</i> ; |
| (4) WWW. | (d) transmiterea scrisorilor electronice; |
| | (e) prelucrarea datelor cu ajutorul unei aplicații de calcul tabelar și transmiterea rezultatelor prin poșta electronică; |
| | (f) prezentarea, transmiterea și căutarea informațiilor în <i>Internet</i> ; |
| | (g) acces la alte calculatoare, calculatorul utilizatorului devenind un terminal al calculatorului aflat la distanță. |

14. Indicați tehnologia de cooperare în rețeaua *Internet*:

- a) de la egal-la-egal;
- b) client-server;
- c) de la egal-la-egal și client-server.

15. Care dintre afirmațiile ce urmează sînt adevărate:

- a) la crearea paginilor Web se folosește limbajul PASCAL;
- b) pentru a trimite o scrisoare electronică, avem nevoie de o conexiune *Internet*;
- c) o pagină Web poate conține numai text și imagini;
- d) la crearea paginilor Web se folosește limbajul HTML;
- e) o pagină Web poate conține text, imagini, secvențe sonore și secvențe video;
- f) scrisorile electronice nu pot fi citite de persoanele străine;
- g) serverul de căutare poate citi informațiile de pe orice calculator conectat la *Internet*?

16. Selectați din lista ce urmează adresele de cutii poștale:

- | | |
|--|---|
| a) <code>www.directie.orhei.md</code> ; | d) <code>bjosu@directie.cahul.md</code> ; |
| b) <code>apetrescu@contabilitate.orhei.md</code> ; | e) <code>www.contabilitate.cahul.md</code> ; |
| c) <code>ivulpe@presedinte.orhei.md</code> ; | f) <code>munteanu@presedinte.calarasi.md</code> . |

RĂSPUNSURI LA TESTELE DE AUTOEVALUARE

Testul nr. 1

1.

```
type SituatieScolara = array [Obiect] of Nota
```

<i>Indicii</i>	Istoria	Geografia	Matematica	Informatica	Fizica
<i>Componente</i>	Nota	Nota	Nota	Nota	Nota

2. Mulțimea de valori ale tipului de date OrarulLectiilor este constituită din tablouri bidimensionale. Liniile sînt specificate prin indici de tipul ZiDeScoa la, iar coloanele – prin indici de tipul Lectie.

3.

- a) `x[1]+ x[2]+ x[3]+ x[4];`
- b) `y[7]+ y[8]+ y[9]+ y[10];`
- c) `abs(x[3]);`
- d) `abs(y[6]);`
- e) `x[1]+ y[10].`

4.

```
Program RTA9;  
{ Raspuns la Testul 1, Itemul 4 }  
type Numere = array [1..50] of integer;  
var A : Numere;  
    n, i : integer;  
begin  
  write('Dati n='); readln(n);  
  { Citim numerele de la tastatura }  
  for i:=1 to n do  
    begin  
      write('A[' , i, ']='); readln(A[i]);  
    end;  
  { Afisam numerele la ecran }  
  writeln('Numerele in ordine inversa:');
```

```

for i:=n downto 1 do
    writeln('A[' , i, ']=', A[i]);
    readln;
end.

```

5. Asupra șirurilor de tip **string** se poate efectua operația de concatenare (juxtapunere), notată prin „+”. Lungimea curentă a unei valori *v* de tip **string** poate fi aflată cu ajutorul funcției predefinite **length(v)** care returnează o valoare de tip **integer**. De asemenea, asupra șirurilor de caractere sînt admise operațiile relaționale <, <=, =, >=, >, <>. Rezultatele acestor operații sînt de tipul **boolean**.

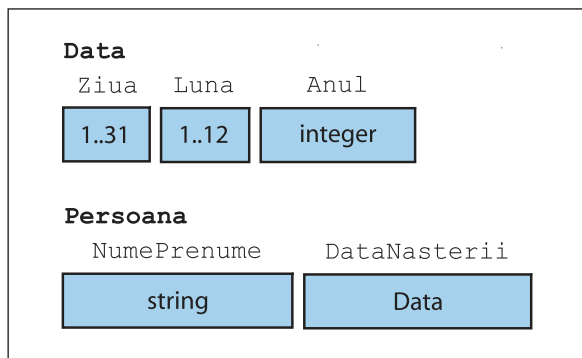
6.

```

Program RTA10;
{ Raspuns la Testul 1, Itemul 6 }
var S : string;
    i : integer;
begin
    writeln('Dati sirul de caractere:');
    readln(S);
    writeln('Sirul in ordine inversa:');
    for i:=length(S) downto 1 do
        write(S[i]);
        writeln;
    readln;
end.

```

7.



8.

```

Program RTA11;
{ Raspuns la Testul 1, Itemul 8 }
type Angajat = record
    NumePrenume : string;
    Salariu : real;
end;
    ListaDePlata = array [1..100] of Angajat;
var L : ListaDePlata;

```

```

    s : real;
    i, n : integer;
begin
    write('Dati n='); readln(n);
    { Citirea datelor despre fiecare angajat }
    for i:=1 to n do
        begin
            writeln('Dati datele despre angajatul ', i);
            write('Numele si prenumele: ');
            readln(L[i].NumePrenume);
            write('Salariul: ');
            readln(L[i].Salariu);
        end;
    { Cautarea salariului maximal }
    s:=0;
    for i:=1 to n do
        if L[i].Salariu > s then s:=L[i].Salariu;
    writeln('Salariul maximal = ', s:10:2);
    { Afisarea informatiilor }
    writeln('Angajatii cu cel mai mare salariu:');
    for i:=1 to n do
        if L[i].Salariu = s then writeln(L[i].NumePrenume);
    readln;
end.

```

9. Instrucțiunea **with** se utilizează pentru a reduce textele programelor PASCAL prin excluderea repetărilor de denumiri ale variabilelor de tip **record** (articol).

10. Valorile variabilei V:

[], ['X'], ['Y'], ['Z'], ['X', 'Y'], ['X', 'Z'], ['Y', 'Z'], ['X', 'Y', 'Z'].

Valorile variabilei I:

[], [8], [9], [8, 9].

11.

```

Program RTA12;
{ Raspuns la Testul 1, Itemul 11 }
var S : string;
    i, n : integer;
begin
    write('Dati un sir de caractere format din ');
    writeln('literele mari ale alfabetului latin:');
    readln(S);
    { Numaram vocalele in sir }
    n:=0;
    for i:=1 to length(S) do
        if S[i] in ['A', 'E', 'I', 'O', 'U'] then n:=n+1;
    writeln('Numarul de vocale = ', n);
    readln;
end.

```

12. Datele fișierelor PASCAL se păstrează pe suporturile de informație ale echipamentelor periferice: discuri și benzi magnetice, discuri optice, hîrtia imprimantei sau a dispozitivului de citit documente ș.a. Procedura assign asociază variabilele de tip *fișier* din componența programelor PASCAL cu fișierele de pe suporturile de informație ale echipamentelor periferice.

13. În funcție de **tipul operațiilor permise** asupra componentelor, fișierele se clasifică în:

- fișiere de intrare (este permisă numai citirea);
- fișiere de ieșire (este permisă numai scrierea);
- fișiere de actualizare (sînt permise scrierea și citirea).

În funcție de **modul de acces** la componente, fișierele se clasifică în:

- fișiere cu acces secvențial sau secvențiale (accesul la componenta i este permis după ce s-a citit/scriș componenta $i - 1$);
- fișiere cu acces aleator sau direct (orice componentă se poate referi direct prin numărul ei de ordine i în fișier).

14.

```
Program RTA13;
{ Raspuns la Testul 1, Itemul 14 }
type Angajat = record
    NumePrenume : string;
    Salariu : real;
end;
FisierAngajati = file of Angajat;
var A : Angajat;
    F : FisierAngajati;
    i, n : integer;
begin
    { Crearea fisierului SALARII.DAT }
    assign(F, 'SALARII.DAT');
    rewrite(F);
    write('Dati n='); readln(n);
    for i:= 1 to n do
        begin
            writeln('Dati datele angajatului ', i);
            write('Nume, prenume: '); readln(A.NumePrenume);
            write('Salariul: '); readln(A.Salariu);
            { Scrierea datelor despre angajat in fisierul F }
            write(F, A);
        end;
    close(F);
end.
```

15.

```
Program RTA14;
{ Raspuns la Testul 1, Itemul 15 }
type Angajat = record
    NumePrenume : string;
    Salariu : real;
end;
FisierAngajati = file of Angajat;
```

```

var A : Angajat;
    F : FisierAngajati;
begin
    assign(F, 'SALARII.DAT');
    reset(F);
    writeln('Datele citite din fisierul SALARII.DAT:');
    while not eof(F) do
        begin
            read(F, A);
            writeln(A.NumePrenume, ' ', A.Salariu:10:2);
        end;
    readln;
    close(F);
end.

```

16.

```

Program RTA15;
{ Raspuns la Testul 1, itemul 16 }
var a, b : real;
    c : string;
    Intrare, Iesire : text;
begin
    assign(Intrare, 'REZULTAT.TXT');
    reset(Intrare);
    assign(Iesire, 'MEDII.TXT');
    rewrite(Iesire);
    while not eof(Intrare) do
        begin
            readln(Intrare, a, b, c);
            writeln(Iesire, (a+b)/2, ' ', c);
            writeln((a+b)/2, ' ', c);
        end;
    readln;
    close(Intrare);
    close(Iesire);
end.

```

Testul nr. 2

1. 000 , 001 , 010 , 011 , 100 , 101 , 110 , 111 .

2. Rezultatul decodificării: CACB. Rezultatul codificării: 00001 00000 00010.

3. Alfabetul latin este format din 26 de litere. Prin urmare, numărul de mesaje posibile ale sursei de informație $n = 26 + 26 = 52$. Cantitatea de informație într-un mesaj:

$$I = \log_2 52 \approx 5,7 \text{ biți.}$$

Din inegalitatea $m \geq I$, obținem $m = 6$.

4. Numărul de mesaje posibile ale calendarului electronic $n = 31 \times 12 \times 100 = 37\,200$.
Cantitatea de informație într-un mesaj:

$$I = \log_2 37\,200 \approx 15,2 \text{ biți.}$$

Din inegalitatea $m \geq I$, obținem $m = 16$.

5. $I = 8$ biți (1 octet).

6. $V = 10 \text{ min.} \times 200 \text{ de caractere/minut} \times 1 \text{ octet} = 2000 \text{ de octeți.}$

7.

```
Program RTA16;  
{ Raspuns la Testul 2, Itemul 7 }  
var c : char;  
begin  
  while not eof do  
    begin  
      readln(c);  
      writeln(c, ' - ', ord(c));  
    end;  
end.
```

8.

```
Program RTA17;  
{ Raspuns la Testul 2, Itemul 8 }  
var i : integer;  
begin  
  while not eof do  
    begin  
      readln(i);  
      writeln(i, ' - ', char(i));  
    end;  
end.
```

9. $I = (10 \times 10) \times 30 \times \log_2 128 = 21\,000 \text{ de biți} = 2625 \text{ de octeți.}$

10. a), c), d), f), h).

$$11. V = 3 \cdot I = 3 \cdot \log_2 \left(\frac{|42 - 34|}{0,1} + 1 \right) \approx 19,02 \text{ biți.}$$

$$12. V = \frac{30 \cdot 60}{3 \cdot 10^{-5}} \log_2 128 = 4,2 \cdot 10^8 \text{ biți} \approx 400,5 \text{ Mbiți.}$$

13. (1) – (e); (2) – (f); (3) – (a); (4) – (c).

14. (1) – (c); (2) – (f); (3) – (a); (4) – (e).

Testul nr. 3

1. a) 667; b) $\approx 31,5926$; c) $\approx 176,7551$.
2. b) 8 nu este o cifră octală; c) 2, 8, 4 și 6 nu sînt cifre binare.
- 3.

```
Program RTA18;
{ Raspuns la Testul 3, Itemul 3 }
var b : integer; { baza }
    x : string; { numarul citit de la tastatura }
    y : integer; { numarul transformat in baza 10 }
    i : integer;
    bi : integer; { baza la puterea i }
begin
    write('Dati baza b='); readln(b);
    write('Dati un numar scris in baza ', b, ': ');
    readln(x);
    y:=0;
    bi:=1; { baza la puterea 0 }
    for i:=length(x) downto 1 do
        begin
            y:=y+(ord(x[i])-48)*bi;
            bi:=bi*b;
        end;
    writeln('Numarul scris in baza 10: ', y);
    readln;
end.
```

4. a) $\approx 1101,1110001$; b) 111011,10110001; c) 10101000,011101001111.
5. a) $\approx 23036,7341$; b) 1333,272; c) 155206,542.
6. a) $\approx 24FF,D611$; b) 2DDB,534; c) 115,7F.
7. $(222221)_4$, $(1000001111)_2$, $(BB)_{16}$, $(132)_8$.
8. $(1D3)_{16} = (723)_8$; $(25)_8 = (21)_{10}$.
- 9.

```
Program RTA19;
{ Raspuns la Testul 3, Itemul 9 }
var x : string; { numarul octal }
    y : string; { numarul binar }
    i : integer;
    T : array ['0'..'7'] of string;
    { T[i] - reprezentarea binara a cifrei octale i }
begin
    { Initializarea tabelului }
    T['0']:= '000'; T['1']:= '001';
    T['2']:= '010'; T['3']:= '011';
    T['4']:= '100'; T['5']:= '101';
```



```

T['6']:= '110'; T['7']:= '111';
write('Dati un numar octal x='); readln(x);
y:='';
for i:=1 to length(x) do
  y:=y+T[x[i]];
writeln('Reprezentarea binara: ', y);
readln;
end.

```

10. a) 1001000111; b) 111100001; c) 100100111; d) 111010.

11.

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

12.

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

13. a) -1; b) +64; c) -39.

14.

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

15. a) $0,10101001 \times 2^5$; b) $-0,100100101 \times 2^7$; c) $-0,11 \times 2^{-3}$.

16. $b = 8$; $x = 25$.

17. a) -26; b) 45; c) -8.

18. a) $[-8, 7]$; b) $[-128, 127]$; c) $[-32\,768, 32\,767]$.

19. a) -0,375; b) 0,125; c) -0,5.

20. $A = -0,9921875$; $B = 0,9921875$.

21. a) -0,109375; b) 7,5; c) -0,25.

22.

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

23. $A = -7,5$; $B = 7,5$.

Testul nr. 4

1.

x	y	z	$\bar{z}$	$y\bar{z}$	$x \vee y\bar{z}$
0	0	0	1	0	0
0	0	1	0	0	0
0	1	0	1	1	1
0	1	1	0	0	0
1	0	0	1	0	1

x	y	z	$\bar{z}$	$y\bar{z}$	$x \vee y\bar{z}$
1	0	1	0	0	1
1	1	0	1	1	1
1	1	1	0	0	1

2. a) și d).

3. {000, 010, 100, 110, 111}.

4.

```

Program RTA20;
{ Raspuns la Testul 4, Itemul 4 }
var x, y, z, f : boolean;
    xx, yy, zz, ff : 0..1;
begin
    write('Dati x='); readln(xx);
    if xx=0 then x:=false else x:=true;
    write('Dati y='); readln(yy);
    if yy=0 then y:=false else y:=true;
    write('Dati z='); readln(zz);
    if zz=0 then z:=false else z:=true;
    f:=x and (not y) or z;
    if f=false then ff:=0 else ff:=1;
    writeln('Valoarea expresiei logice: ', ff);
    readln;
end.

```

5.

```

Program RTA21;
{ Raspuns la Testul 4, Itemul 5 }
var x, y, z, f : boolean;
    xx, yy, zz, ff : 0..1;
begin
    writeln('x y z f');
    writeln('——');
    for x:=false to true do
        for y:=false to true do
            for z:=false to true do
                begin
                    f:=(not x) and (not y) or z;
                    if x=false then xx:=0 else xx:=1;
                    if y=false then yy:=0 else yy:=1;
                    if z=false then zz:=0 else zz:=1;
                    if f=false then ff:=0 else ff:=1;
                    writeln(xx, ' ', yy, ' ', zz, ' ', ff);
                end;
            end;
        end;
    end;
    readln;
end.

```

6. Domeniul de definiție: {000, 001, 010, 011, 100, 101, 110, 111}. Domeniul de valori: {0, 1}.

7.

x_1	x_2	x_3	y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

8.

```
Program RTA22;
{ Raspuns la Testul 4, Itemul 8 }
var x1, x2, x3, y : boolean;
    xx1, xx2, xx3, yy : 0..1;
begin
  write('Dati x1='); readln(xx1);
  if xx1=0 then x1:=false else x1:=true;
  write('Dati x2='); readln(xx2);
  if xx2=0 then x2:=false else x2:=true;
  write('Dati x3='); readln(xx3);
  if xx3=0 then x3:=false else x3:=true;
  y:=x1 and ((not x2) or (not x3));
  if y=false then yy:=0 else yy:=1;
  writeln('Valoarea functiei logice y=', yy);
  readln;
end.
```

9.

```
Program RTA21;
{ Raspuns la Testul 4, Itemul 9 }
var x1, x2, x3, y : boolean;
    xx1, xx2, xx3, yy : 0..1;
begin
  writeln('x1 x2 x3 y');
  writeln('———');
  for x1:=false to true do
    for x2:=false to true do
      for x3:=false to true do
        begin
          y:=(not x1) and (x2 or x3);
          if x1=false then xx1:=0 else xx1:=1;
          if x2=false then xx2:=0 else xx2:=1;
          if x3=false then xx3:=0 else xx3:=1;
```

```

    if y=false then yy:=0 else yy:=1;
    writeln(' ', xx1, ' ', xx2, ' ', xx3, ' ', yy);
end;
readln;
end.

```

10. $2^{25} = 4294967296$.

11.

SI-NU

x_1	x_2	$\overline{x_1 x_2}$
0	0	1
0	1	1
1	0	1
1	1	0

SAU-NU

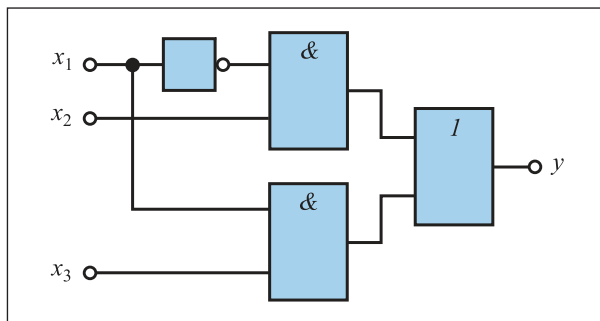
x_1	x_2	$\overline{x_1 \vee x_2}$
0	0	1
0	1	0
1	0	0
1	1	0

12. a) și c); b) și d).

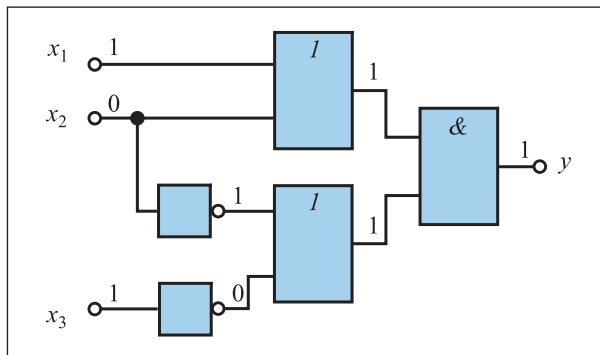
13. b).

Testul nr. 5

1.



2.



$$3. y = (x_1 \vee x_2) (\bar{x}_2 \vee \bar{x}_3).$$

4.

```

Program RTA24;
{ Raspuns la Testul 5, Itemul 4 }
var a, b, s, t : char;
begin
  write('Dati a='); readln(a);
  write('Dati b='); readln(b);
  if (a='0') and (b='0') then begin s:='0'; t:='0'; end;
  if (a='0') and (b='1') then begin s:='1'; t:='0'; end;
  if (a='1') and (b='0') then begin s:='1'; t:='0'; end;
  if (a='1') and (b='1') then begin s:='1'; t:='1'; end;
  writeln('s=', s, ' ', 't=', t);
  readln;
end.

```

5. Un semisumator conține 2 porți logice *NU*, 3 porți logice *ȘI* și o poartă logică *SAU* (fig. 5.10). Un sumator elementar conține 2 semisumatoare și o poartă logică *SAU* (fig. 5.11). Prin urmare, un sumator elementar conține 4 porți logice *NU*, 6 porți logice *ȘI* și 3 porți logice *SAU*. Evident, un sumator de n biți conține $4n$ porți logice *NU*, $6n$ porți logice *ȘI* și $3n$ porți logice *SAU* (fig. 5.12). Pentru un sumator de $n = 8$ biți, obținem: 32 de porți logice *NU*; 48 de porți logice *ȘI*; 24 de porți logice *SAU*.

6.

```

Program RTA25;
{ Raspuns la Testul 5, Itemul 6 }
label 1;
var A, B, S : string;
    t : char; { transportul }
    i : integer;
begin
  writeln('Dati doua numere binare A si B. Fiecare numar');
  writeln('trebuie sa contina exact 8 cifre binare. ');
  write('A='); readln(A);
  write('B='); readln(B);
  t:='0'; { initial transportul t=0 }
  S:='xxxxxxxx'; { initial suma S este necunoscuta }
  for i:=8 downto 1 do
    begin
      { adunarea cifrelor binare A[i], B[i], T[i-1] }
      if (A[i]='0') and (B[i]='0') and (t='0') then
        begin S[i]:='0'; t:='0'; goto 1; end;
      if (A[i]='0') and (B[i]='0') and (t='1') then
        begin S[i]:='1'; t:='0'; goto 1; end;
      if (A[i]='0') and (B[i]='1') and (t='0') then
        begin S[i]:='1'; t:='0'; goto 1; end;
      if (A[i]='0') and (B[i]='1') and (t='1') then
        begin S[i]:='0'; t:='1'; goto 1; end;
    
```

```

    if (A[i]='1') and (B[i]='0') and (t='0') then
    begin S[i]:='1'; t:='0'; end;
    if (A[i]='1') and (B[i]='0') and (t='1') then
    begin S[i]:='0'; t:='1'; goto 1; end;
    if (A[i]='1') and (B[i]='1') and (t='0') then
    begin S[i]:='0'; t:='1'; goto 1; end;
    if (A[i]='1') and (B[i]='1') and (t='1') then
    begin S[i]:='1'; t:='1'; goto 1; end;
1: end;
writeln('S=', S);
writeln('Cifra de depasire t=', t);
readln;
end.

```

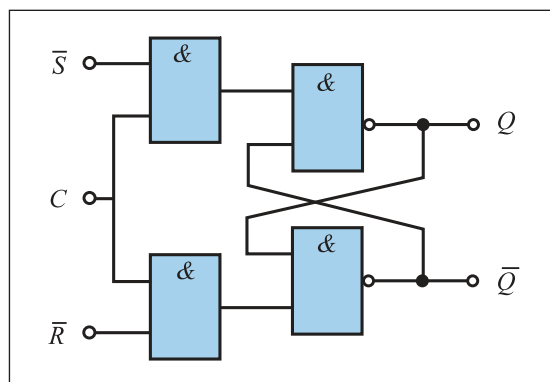
7. (1) – (e); (2) – (g); (3) – (a); (4) – (d); (5) – (h); (6) – (c).

8. y_1 – SUS; y_2 – JOS; y_3 – STÎNGA; y_4 – DREAPTA.

x_1	x_2	y_1	y_2	y_3	y_4
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

9. $Q = 1$ și $\bar{Q} = 0$.

10.



11. (1) – (d); (2) – (i); (3) – (g); (4) – (a); (5) – (c); (6) – (f).

12. 00010011.

13. 11010000.

14.

```

Program RTA26;
{ Raspuns la Testul 5, Itemul 14 }
var R : string;
    n, m, i, j : integer;

```

```

begin
  writeln('Dati continutul initial al registrului');
  write('R='); readln(R);
  n:=length(R); { capacitatea registrului }
  write('Dati m='); readln(m);
  for j:=1 to m do
    begin
      { deplasarea cu o pozitie }
      for i:=1 to n-1 do R[i]:=R[i+1];
      R[n]:='0';
    end;
  writeln('Continutul registrului dupa deplasare');
  writeln('R=', R);
  readln;
end.

```

15. 10010011.

16. 10101001.

17.

```

Program RTA27;
{ Raspuns la Testul 5, Itemul 17 }
label 1;
var A : string;
      n, m, i, j : integer;
      t : char; { transportul }
begin
  writeln('Dati continutul initial al numaratorului');
  write('A='); readln(A);
  n:=length(A); { capacitatea numaratorului }
  write('Dati m='); readln(m);
  for j:=1 to m do
    begin
      { adunarea unitatii la continutul numaratorului }
      t:='1';
      for i:=n downto 1 do
        begin
          { adunarea transportului la cifra A[i] }
          if (A[i]='0') and (t='0') then
            begin A[i]:='0'; t:='0'; goto 1; end;
          if (A[i]='0') and (t='1') then
            begin A[i]:='1'; t:='0'; goto 1; end;
          if (A[i]='1') and (t='0') then
            begin A[i]:='1'; t:='0'; goto 1; end;
          if (A[i]='1') and (t='1') then
            begin A[i]:='0'; t:='1'; goto 1; end;
          1: end;
        end;
      writeln('B=', A);
      readln;
    end.

```

Testul nr. 6

1. (1) – (i); (2) – (f); (3) – (a); (4) – (c); (5) – (d); (6) – (g).

2. Tastatura, mouse-ul, vizualizatorul, imprimanta, unitatea de disc magnetic, unitatea de disc optic, cititorul de documente (scannerul).

3. a), c), f).

4. Magistrala asigură comunicarea procesorului cu memoria internă și echipamentul periferic.

5. Vezi figura 6.2. Componentele obligatorii: procesorul, magistrala, memoria internă, controlerul, dispozitivul de intrare-ieșire.

6. *Codul instrucțiunii* – în acest câmp se indică operația ce trebuie executată. *Adresă operand 1* și *Adresă operand 2* – în aceste câmpuri se indică adresele locațiilor din memoria internă care conțin, respectiv, primul și al doilea operand. *Adresă rezultat* – indică adresa locației din memoria internă în care va fi depus rezultatul operației.

7. a) numărul din locația 101 este adunat cu numărul din locația 153, iar rezultatul obținut este depus în locația 342;

b) numărul din locația 508 este împărțit la numărul din locația 391, iar rezultatul obținut este depus în locația 216;

c) numărul din locația 751 este înmulțit cu numărul din locația 852, iar rezultatul obținut este depus în locația 031;

d) din numărul din locația 450 se scade numărul din locația 709, iar rezultatul obținut este depus în locația 011.

8. a), c), f) – instrucțiuni operaționale; g) – instrucțiune de transfer; e) – instrucțiune de salt; b), d) – instrucțiuni de intrare-ieșire.

9.

INC	X
MEM	S
SCD	Y
ADU	X

10.

01	583
02	461
01	971
02	583
01	461

11. a), d), f), h) – resurse tehnice; b), c), e), g) – resurse programate.

12. În cazul memoriilor externe cu acces secvențial scrierea/citirea înregistrării dorite este posibilă doar după scrierea/citirea tuturor înregistrărilor precedente. În cazul memoriilor externe cu acces direct orice înregistrare poate fi scrisă/citită fără a mai citi/scrie înregistrările precedente.

13. b), c), e).

14. b), e).

15. 2867,2 Kocteți/s.

16. a), c), d).

17. a), b), e).

18. Parametrii de bază ai unui calculator:

- viteza de operare;
- capacitatea memoriei interne;
- componența, capacitatea și timpul de acces ale unităților de memorie externă;
- componența și parametrii tehnici respectivi ai echipamentelor periferice;
- parametrii de masă și gabarit;
- costul.

19. Parametrii de bază ai microprocesorului:

- lungimea cuvântului;
- frecvența ceasului de sistem;
- capacitatea magistralelor;
- capacitatea de prelucrare.

Testul nr. 7

1. a) ≈ 6 ore; b) 12 min. 30 s; c) $\approx 0,73$ s; d) $\approx 0,00072$ s.

2. b), d).

3. Calculatoarele, adaptoarele de rețea, structura de comunicare.

4. a), c).

5. a), d).

6. Un protocol de rețea definește modul de adresare a calculatoarelor, lungimea și componența pachetelor de date, algoritmul de depistare și corectare a erorilor, modul de conectare fizică a adaptoarelor și cablurilor de rețea.

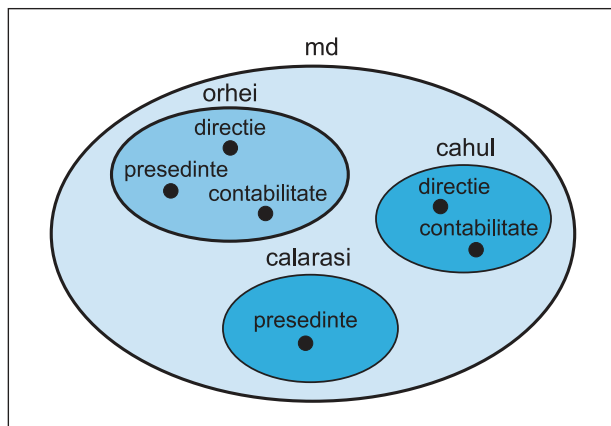
7. Arhitectura unei rețele reprezintă modul în care este concepută rețeaua: topologia, protocoalele de comunicație, tehnologiile de cooperare în rețea.

8. Vezi figura 7.5.

9. d).

10. În *Internet* calculatoarele se identifică cu ajutorul adreselor. Acestea pot fi de două tipuri: adrese numerice și adrese simbolice.

11. Întrucât $(214)_{10} = (11010110)_2$, iar biții 0-2 au valoarea 110, adresa *Internet* aparține clasei C. În cazul clasei C adresa subrețelei se indică în biții 3-32 ai adresei numerice. Din primul octet al adresei numerice selectăm biții 3-7 și transformăm numărul binar obținut în sistemul zecimal: $(10110)_2 = (22)_{10}$. Prin urmare, adresa subrețelei este 22.121.216. Numărul 109 din ultimul octet al adresei numerice reprezintă adresa calculatorului în subrețea.



12. md – domeniul de cel mai înalt nivel. Acest domeniu include subdomeniile **orhei**, **cahul** și **calarasi**. Subdomeniul **orhei** include calculatoarele **directie**, **contabilitate** și **presedinte**. Subdomeniul **cahul** include calculatoarele **directie** și **contabilitate**. Subdomeniul **calarasi** include calculatorul **presedinte**.

13. (1) – (g); (2) – (a); (3) – (d); (4) – (f).

14. b).

15. b), d), e).

16. b), c), d), f).

Bibliografie

1. Bolun Ion. *Inițiere în rețele. INTERNET*. Chișinău, Editura ASEM, 1997.
2. Ceapăru Mihai. *Comunicația prin intermediul rețelelor de calculatoare*. București, Editura Tehnică, 1996.
3. Cerchez Emanuela. *Internet. Manual pentru liceu. Filiera teoretică*. Iași, Editura Polirom, 2000.
4. Cerchez Emanuela, Șerban Marinell. *Informatica pentru gimnaziu*. Iași, Editura Polirom, 2002.
5. Gremalschi Anatol, Mocanu Iurie, Spinei Ion. *Informatica. Limbajul PASCAL*. Chișinău, Editura Știința, 2003.
6. Gremalschi Ludmila, Mocanu Iurie. *Structura și funcționarea calculatorului. Material didactic pentru licee și colegii*. Chișinău, Editura Lyceum, 1996.
7. Gremalschi Anatol, Bejan Viorel, Gremalschi Ludmila. *Structura calculatoarelor numerice. Material didactic*. Chișinău, UTM, 1996.
8. Levine John R. *Internet pentru toți*. București, Editura Teora, 1996.
9. Lowe Doug. *Rețele pentru toți*. București, Editura Teora, 1995.
10. Mârșanu Radu. *Sisteme de calcul. Manual pentru licee de informatică, clasa a IX-a*. R.A. București, Editura Didactică și Pedagogică, 1995.
11. Mârșanu Radu, Velicanu Manole. *Tehnică de calcul. Manual pentru clasa a XII-a*. București, Editura ALL, 1999.
12. Mihoc Dan, Iliescu Sergiu Stelian. *Elemente de informatică. Mecanizarea și automatizarea producției. Manual pentru licee industriale, clasa a XII-a*. R.A. București, Editura Didactică și Pedagogică, 1995.
13. Mucenic Băsoiu, Mihai Băsoiu, Eugen Ștefan. *Compact disc*. București, Editura Teora, 1995.
14. Pfaffenberger Bryan, Petersen Judy. *Dicționar explicativ de calculatoare*. București, Editura Teora, 1996.
15. Petrescu Adrian, Iacob Francisc, Racoviță Zoe. *Inițiere în structura calculatoarelor electronice*. București, Editura Teora, 1996.
16. Petrescu Silviu. *Informatica aplicată: manual de informatică pentru clasa a XII-a*. București, Editura Teora, 1998.
17. Secrieru Nicolae, Gremalschi Anatol, Cornea Ion. *Arhitectura și organizarea microprocesoarelor*. Chișinău, Editura Universitas, 1995.
18. Velicanu Manole, Vasilescu Adrian. *Bazele informaticii. Manual pentru clasa a XII-a*. București, Editura ALL, 1999.
19. Залогова Л.А., Плаксин М.А., Русаков С.В., Русакова О.Л. и др. *Информатика. Задачник-практикум в 2 т.* / Под ред. Семакина И.Г., Хеннера Е.К.: Том 1. – М.: Лаборатория Базовых Знаний, 1999.
20. Семакин И.Г., Залогова Л.А., Русаков С.В., Шестакова Л.В. *Информатика. Базовый курс для 7-9 классов*. – М.: Лаборатория Базовых Знаний, 1999.

Manualul acesta este proprietatea Ministerului Educației al Republicii Moldova.

Liceul _____				
Manualul nr. _____				
Anul de folosire	Numele și prenumele elevului	Anul școlar	Aspectul manualului	
			la primire	la restituire
1.				
2.				
3.				
4.				
5.				

- Dirigintele trebuie să controleze dacă numele elevului este scris corect.
- Elevul nu trebuie să facă niciun fel de însemnări pe pagini.
- Aspectul manualului (la primire și la restituire) se va aprecia folosind termenii: *nou, bun, satisfăcător, nesatisfăcător*.