BAZELE INFORMATICII
ELEMENTE DE PROGRAMARE C/C++

Asist. univ.dr.
Aurelia PĂTRAȘCU
Universitatea Petrol–Gaze din Ploiești

Prof. univ. dr.
Marian ZAHARIA
Universitatea Petrol–Gaze din Ploiești

Lect. univ. dr.
Ana TĂNĂSESCU
Universitatea Petrol–Gaze din Ploiești

Aurelia PĂTRAȘCU

Marian ZAHARIA **Ana TĂNĂSESCU**

BAZELE INFORMATICII
ELEMENTE DE PROGRAMARE C/C++

EDITURA UNIVERSITARĂ
București, 2011

Coperta: Angelica Mălăescu

Editura Universitară
Director: Drd. Vasile Muscalu
B-dul. N. Bălcescu nr. 27-33, Sector 1, București
Tel.: 021 – 315.32.47 / 319.67.27
www.editurauniversitara.ro
e-mail: redactia@editurauniversitara.ro

EDITURĂ RECUNOSCUTĂ DE CONSILIUL NAȚIONAL AL CERCETĂRII ȘTIINȚIFICE DIN ÎNVĂȚĂMÂNTUL SUPERIOR (C.N.C.S.I.S.)

Descrierea CIP a Bibliotecii Naționale a României
PĂTRAȘCU, AURELIA
Bazele informaticii : elemente de programare C/C++ /
Pătrașcu Aurelia, Zaharia Marian, Tănăsescu Ana. - București :
Editura Universitară, 2011
Bibliogr.
ISBN 978-606-591-328-8

I. Zaharia, Marian
II. Tănăsescu, Ana

004.43 C
004.43 C++

Distribuție: tel.: 021-315.32.47 /319.67.27,
comenzi@editurauniversitara.ro

ISBN 978-606-591-328-8

DOI: 10.5682/9786065913288

PREFAȚĂ

După cum rezultă şi din titlul lucrării, scopul principal al acesteia este de inițiere în tainele informaticii şi programării structurate în limbajul C/C++. Din acest considerent lucrarea se opreşte numai asupra unor capitole care să ajute cititorii sa facă primii paşi în programarea structurată. Ea va fi urmată de o altă lucrare a noastră care va viza programarea avansată în C/C++, unde vor fi abordate problematicile construirii şi utilizării funcțiilor, declararea şi utilizarea pointerilor către funcții, crearea şi exploatarea structurilor autoreferite, precum şi bazele programării orientate obiect.

Lucrarea este structurată în nouă capitole, însoțite de numeroase aplicații practice, scheme şi comentarii, şi abordează, gradat, conceptele şi elementele fundamentale ale definirii algoritmilor şi schemelor logice, precum şi ale implementării acestora în limbajul C/C++.

Capitolul I, ***Arhitectura sistemelor de calcul şi bazele aritmetice şi logice ale calculatoarelor***, aduce în atenția cititorului elementele structurale fundamentale ale calculatoarelor numerice. De asemenea, se prezintă, bazat pe exemple concrete, bazele aritmetice şi logice ale acestora.

Capitolul II, ***Algoritmi şi scheme logice***, oferă cititorului un set de cunoştințe de bază privind construirea algoritmilor, precum şi premisele utilizării acestora la modelarea structurilor fundamentale ale programării structurate.

Capitolul III, ***Prezentarea generală a limbajului C***, are rolul de a ajuta utilizatorul să facă primii paşi în cunoaşterea şi utilizarea limbajului C/C++, în conceperea şi dezvoltarea aplicațiilor sale. Sunt trecute în revistă elementele fundamentale de sintaxă, pornind de la alfabet şi semne de punctuație, până la scrierea propozițiilor şi/sau frazelor, precum şi funcții de intrare/ieşire necesare construirii unor interfețe minime între utilizator şi aplicațiile sale. De asemenea, sunt prezentate directivele procesor *define* şi *include,* acestea fiind unele dintre cele mai utilizate directive procesor la dezvoltarea de aplicații C/C++, directive folosite în exemplele prezentate în lucrare.

Capitolul IV, ***Tipuri fundamentale de date şi clase de memorie***, abordează probleme privind declararea şi inițializarea tipurilor fundamentale de date, modurile de reprezentare ale acestora, domeniile valorilor

reprezentabile, precum şi modul de vehiculare a datelor de diverse tipuri în cadrul aplicațiilor.

Capitolul V, ***Operatori şi expresii***, prezintă clasele de operatori ai limbajului C/C++, modurile lor de aplicare, efectele produse asupra datelor implicate, precum şi modurile de evaluare ale expresiilor în care apar.

Capitolul VI, ***Intrucțiuni***, abordează structurile de programare fundamentale (secvențială, alternativă şi repetitivă), instrucțiunile care permit construirea şi exploatarea acestor structuri, instrucțiunile de salt şi de inserare în codul sursă C/C++ a instrucțiunilor în limbaj de asamblare.

Capitolul VII, ***Pointeri***, este dedicat prezentării modurilor de declarare şi de încărcare a pointerilor, inclusiv alocarea dinamică de memorie, trecerii în revistă a principalelor operații care pot fi efectuate cu pointeri din aşa-numita aritmetică de pointeri. Ținând seama de importanța înțelegerii şi dobândirii de deprinderi în lucrul cu pointeri, în cadrul capitolului sunt prezentate o serie de exemple edificatoare.

Capitolul VIII, ***Masive***, este destinat prezentării modurilor de definire şi utilizare a masivelor de date în C/C++. Sunt abordate tematici privind declararea masivelor, inițializarea acestora la declarare, prin încărcare individuală sau citirea de la *stdin*, referirea elementelor masivelor prin indexare şi prin indirectare, lucrul cu masive de pointeri şi alocarea dinamică de memorie pentru masive.

Capitolul IX, ***Caractere şi şiruri de caractere***, după trecerea în revistă a unor aspecte specifice lucrului cu caractere şi masive de caractere, precum şi a funcțiilor şi macrodefinițiilor pentru lucrul cu acestea, prezintă modurile de definire şi inițializare a şirurilor de caractere, funcțiile specifice de intrare/ieşire, precum şi o gamă largă de funcții destinate lucrului cu şiruri de caractere. În scopul facilitării parcurgerii şi însuşirii cunoştințelor privind lucrul cu caractere şi şiruri de caractere, pe parcursul capitolului sunt prezentate aplicații însoțite de comentarii.

Conştienți că lucrarea este perfectibilă, mulțumim anticipat celor care, prin observațiile şi recomandările lor, vor contribui la îmbunătățirea acesteia.

Autorii

CUPRINS

Capitolul 1

Arhitectura sistemelor de calcul şi bazele aritmetice şi logice ale calculatoarelor

Informatica reprezintă un complex de discipline prin care se asigură prelucrarea raţională a informaţiilor prin intermediul maşinilor automate.

Prelucrarea automată a informaţiei s-a realizat o dată cu apariţia calculatoarelor electronice. Aşadar, scopul utilizării unui calculator este de a prelucra informaţia. Informaţia prelucrată poate fi formată din texte, numere, imagini sau sunete şi este păstrată pe diferite medii de memorare, în diferite formate, sub formă de date.

1.1 Structura unui calculator electronic

Structura unui calculator numeric a fost definită în anul 1945 de către matematicianul John von Neumann în lucrarea "Prima schiţă de Raport asupra EDVAC" şi este prezentată în figura 1.1 [1, 2, 10, 11, 26, 28].

În această figură, liniile continue reprezintă fluxuri de date, liniile întrerupte reprezintă fluxuri de comenzi şi de stări, iar dreptunghiurile - blocuri funcţionale. Blocurile funcţionale, simbolurile acestora şi destinaţia lor sunt următoarele:

- ✓ *unitatea de intrare* (**UI**) este destinată introducerii de date şi programe în sistemul de calcul (tastatură, scanner, joystick, mouse etc.);
- ✓ *memoria* (**M**) permite stocarea datelor şi a programelor în scopul prelucrării;
- ✓ *unitatea aritmetico-logică* (**UAL**) efectuează calcule aritmetice şi operaţii logice;

- ✓ *unitatea centrală* (**UC**) este componenta care coordonează întreaga activitate din sistemul de calcul;
- ✓ *unitatea de ieşire* (**UE**) permite transmiterea rezultatelor obținute prin prelucrarea datelor de intrare către utilizator. Extragerea datelor şi a rezultatelor se face prin utilizarea unor dispozitive de ieşire printre care: monitorul, imprimanta, plotter-ul, boxe etc. Imprimanta transferă informațiile pe suport de hârtie, fiind, la rândul ei, de diverse tipuri (matricială, cu jet de cerneală, laser etc.)

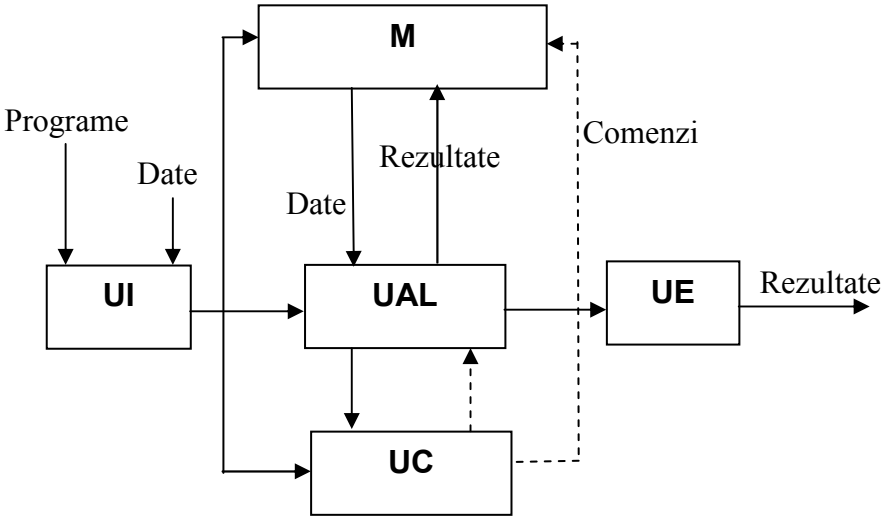


Figura 1.1. *Schema von Neumann a unui calculator numeric* [10, 11].

Dintre blocurile menționate în figura 1.1., din punct de vedere al funcțiilor sale, cel mai important este unitatea centrală (UC). Aceasta are în structura sa două unități care îi asigură funcționalitatea:

- **unitatea de memorie internă** este destinată, pe de o parte, păstrării informațiilor de configurare, iar, pe de altă parte, păstrării datelor şi instrucțiunilor programelor aflate în execuție. Aceste funcții sunt asigurate de:
  - memoria Read Only Memory **(ROM)** aflată într-un cip pe placa de bază şi utilizată pentru a stoca informații despre hardware şi anumite programe care configurează diverse dispozitive ale calculatorului, lansate în execuție la pornirea calculatorului. Memoria ROM ale următoarele caracteristici:
    - nu îşi pierde conținutul la oprirea calculatorului;

  - nu poate fi scrisă de către utilizatorul calculatorului;
  - este inscripționată de către producător cu ajutorul unei aparaturi speciale.
- memoria Random Access Memory (**RAM**) care stochează temporar programele în timp ce acestea rulează, împreună cu datele utilizate de aceste programe. Prin lansarea în execuție a unui program, fișierele sunt aduse în memoria RAM. Acestea rezidă în memoria RAM, atâta timp cât rulează.
  Memoria RAM se caracterizează prin:
  - este o memorie volatilă (la închiderea calculatorului tot ce este în memoria RAM se pierde);
  - fizic, este un ansamblu de *cipuri* sau de module conținând cipuri care sunt introduse într-un conector de pe placa de bază.

- **unitatea centrală de prelucrare** sau **procesorul (central processing unit).** Procesorul este montat în interiorul calculatorului pe placa[1] de bază, fiind cel mai important *cip* din sistemul de calcul. Dintre funcțiile procesorului se menționează:
  - realizează calculele şi operaţiile logice;
  - execută instrucțiuni pentru programe;

[1] Placa de bază (*Motherboard*) este cea mai importantă componentă a sistemului de calcul. Pe ea sunt montate toate componentele care controlează sistemul de calcul. Dintre cele mai cunoscute se amintesc:

- soclul (conectorul) procesorului;
- soclurile de memorie (RAM);
- cipul BIOS;
- setul de cipuri al plăcii de bază;
- conectoarele de magistrală:
  - conectoare ISA (Industry Standard Architecture);
  - conectoare PCI (Peripheral Component Interconnect);
  - conectoare AGP (Accelerated Graphics Port).
- cipurile super I / O:
  - controlere pentru unitățile de dischetă;
  - controlere pentru porturi paralele;
  - controlere pentru porturi seriale;
  - controlere pentru tastatură şi mouse.

- controlează operațiile efectuate de alte componente ale calculatorului.

Unitatea centrală şi unitatea aritmetico-logică prelucrează doar valori numerice în format binar (numere în baza 2). Aceste valori sunt greu de manevrat de către un utilizator obişnuit. Din acest motiv, pentru reprezentarea informațiilor prelucrate, se folosesc diverse modalități de codificare, în funcție de tipul valorilor pe care le iau datele respective.

1.2 Sisteme de numerație

Un *sistem de numerație* este format din totalitatea regulilor de reprezentare a numerelor, folosind un anumit set de simboluri distincte, numite *cifre* [1, 2, 10, 29].

Sistemele de numerație sunt de două tipuri:

- sisteme de numerație poziționale;
- sisteme de numerație nepoziționale.

Sistemele de numerație poziționale sunt acele sisteme de numerație pentru care valoarea unei cifre din cadrul unui număr depinde de poziția ocupată de acea cifră în cadrul numărului. Sistemul arab zecimal de numerație este unul pozițional. De exemplu, cifra 8 are valoarea 80 în numărul 53986 şi are valoarea 80000 în numărul 85712.

Sistemele de numerație nepoziționale sunt sistemele de numerație pentru care valoarea unei cifre dintr-un număr nu este unic determinată de poziția cifrei în număr, ci de contextul în care se află cifra. De exemplu, sistemul roman de numerație este unul nepozițional. Astfel, valoarea cifrei I în numărul VI este +1, iar în numărul IV este –1.

Baza unui sistem de numerație pozițional este dată de numărul de elemente care formează alfabetul sistemului de numerație. Se consideră că alfabetul este format din cifre care sunt numere întregi, consecutive, nenegative. De exemplu, sistemul de numerație în baza 2 are alfabetul {0,1}, sistemul de numerație în baza 10 are alfabetul {0,1,2,3,4,5,6,7,8,9}, iar sistemul de numerație în baza 16 are alfabetul {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F}.

1.2.1 Conversia dintr-o bază oarecare B în baza 10

Fie sistemul de numerație în baza B, cu alfabetul {0,1,...,B-1}. Un număr rațional pozitiv care este reprezentat în baza B prin şirul:

$$Q_B = b_{n-1}b_{n-2}\ldots b_0, b_{-1}\ldots b_{-m} \tag{1.1}$$

are, prin definiție, următoarea valoare în baza 10:

$$Q_{10} = b_{n-1}\cdot B^{n-1} + b_{n-2}\cdot B^{n-2} + \ldots + b_1\cdot B + b_0 + b_{-1}\cdot B^{-1} + \ldots b_{-m}\cdot B^{-m} \tag{1.2}$$

De exemplu, numărul $81A_{16}$, în baza 10 devine :

$$81A_{16} = 8\cdot 16^2 + 1\cdot 16^1 + A\cdot 16^0 = 2048 + 16 + 10 = 2074_{10}$$

iar numărul $11{,}101_2$ devine :

$$11{,}101_2 = 1\cdot 2^1 + 1\cdot 2^0 + 1\cdot 2^{-1} + 0\cdot 2^{-2} + 1\cdot 2^{-3} = 3{,}625_{10}$$

Se observă că prin conversia părții întregi a numărului: $b_{n-1}b_{n-2}\ldots b_0$ se obține un întreg, iar prin conversia părții fracționare: $b_{-1}b_{-1}\ldots b_{-m}$ se obține un număr subunitar.

1.2.2 Conversia din baza 10 într-o bază de numerație oarecare B

Conversia unui număr din baza 10 într-o bază oarecare B se realizează conform următorului algoritm:

- ✓ se convertește *partea întreagă* a numărului prin împărțiri întregi, succesive la baza B:
  - în urma fiecărei împărțiri se obține un nou cât şi un rest;
  - noul cât este deîmpărțitul următoarei împărțiri întregi;
  - algoritmul se încheie când se obține câtul 0;

  - resturile obținute, începând cu ultimul şi până la primul, reprezintă cifrele numărului, de la cea mai semnificativă la cea mai puțin semnificativă.

- ✓ se convertește *partea fracționară* a numărului prin înmulțiri succesive, cu baza B:
  - partea fracționară a fiecărui produs constituie deînmulțitul pentru produsul următor;
  - partea fracționară a numărului convertit în baza B este reprezentată de succesiunea obținută din părțile întregi ale tuturor produselor obținute, începând cu primul produs care furnizează cifra cea mai semnificativă a rezultatului;
  - algoritmul se încheie cu un rezultat exact, atunci când se obține ca produs parțial un întreg;
  - algoritmul se încheie cu o aproximare a numărului fracționar, după un anumit număr de paşi, atunci când nu se poate obține ca produs parțial un întreg.

Exemplul 1. Conversia numărului 2536 în baza 5 se realizează astfel:

2536 : 5 = 507 rest 1
507 : 5 = 101 rest 2
101 : 5 = 20 rest 1
20 : 5 = 4 rest 0
4 : 5 = 0 rest 4

Rezultatul obținut este $2536_{10}=40121_5$.

Exemplul 2. Conversia numărului 0,208984375 în baza 8 presupune efectuarea următoarelor etape:

0,208984375*8=1,671875
0,671875 *8=5,375
0,375 *8=3

Rezultatul conversiei este $0,153_8$.

1.3 Sistemul binar

Acest subcapitol este destinat prezentării modului de conversie din sistemul de numerație zecimal în cel binar şi invers, precum şi a efectuării operațiilor aritmetice în sistemul de numerație binar.

1.3.1 Conversia numerelor din sistemul binar în cel zecimal şi invers

Trecerea de la baza 2 la baza 10 se face pornind de la exprimarea numărului binar în funcție de puteri ale lui 2. De exemplu:

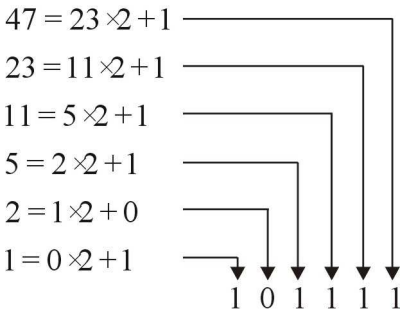
$101101 = 1\cdot 2^5 + 0\cdot 2^4 + 1\cdot 2^3 + 1\cdot 2^2 + 0\cdot 2^1 + 1\cdot 2^0 = 1\cdot 32 + 0\cdot 16 + 1\cdot 8 + 1\cdot 4 + 0\cdot 2 + 1\cdot 1 =$
$= 45 \Rightarrow 101101_2 = 45_{10}$ indicele fiind baza în care este scris numărul.

Pentru a trece un număr din baza 10 în baza 2, se consideră numărul natural ***n*** scris în baza 10. Pentru a afla corespondentul binar al acestuia, se notează cu $c_1, c_2, ..., c_k$ câturile şi cu $r_1, r_2, ..., r_k$ resturile respective ale împărțiri succesive la 2. Deoarece împărțitorul este 2, resturile vor fi 0 sau 1. Împărțirile la 2 se continuă până când $c_k = 0$. Astfel, se obține următorul şir de egalități:

$$\begin{aligned} c_1 &= c_2 \cdot 2 + r_2 \\ c_2 &= c_3 \cdot 2 + r_3 \\ &\vdots \quad \vdots \quad \vdots \quad \vdots \\ c_{k-1} &= c_k \cdot 2 + r_k \\ &\text{unde } c_k = 0. \end{aligned}$$

Numărul binar corespunzător lui ***n*** este: $r_k r_{k-1} ... r_2 r_1$.

De exemplu, pentru ***n*** = 47 se obține:



adică $47_{10} = 101111_2$, iar $k = 6$, adică $c_6 = 0$.

1.3.2 Operații aritmetice în sistemul binar

În continuare, se vor prezenta cele patru operații aritmetice elementare: adunarea, scăderea, înmulțirea şi împărțirea.

❒ Adunarea

Operația de adunare în sistemul binar se desfăşoară conform următoarei tabele:

$$\begin{cases} 0+0=0 \\ 0+1=1 \\ 1+0=1 \\ 1+1=10 \end{cases}$$

În situația 1 + 1 = 10, se scrie la rezultat cifra 0 şi se păstrează 1 pentru poziția de ordin superior (caz analog cu 5 + 5 = 10 în baza 10, când se scrie 0 şi se păstrează 1).

În cazul 1 + 1 + 1 = 10 + 1 = 11, se scrie 1 şi se păstrează 1 pentru poziția de ordin superior.

Cifrele care se păstrează pentru poziția de ordin superior se numesc cifre de transport.

Exemplul 1: x = 11101 şi y = 1101; x + y = ?

$$\begin{array}{r} 11101+ \\ 1101 \\ \hline 101010 \end{array}$$

Exemplul 2: x = 1100011011 şi y = 101111101; x + y = ?

$$\begin{array}{r} \underline{111111111} \rightarrow \text{cifrele de transport} \\ 1100011011+ \\ 101111101 \\ \hline 10010011000 \end{array}$$

❒ Scăderea

Tabela operației de scădere în sistem binar este:

$$\begin{cases} 0-0=0 \\ 0-1=\text{X} \\ 1-0=1 \\ 1-1=0 \end{cases}$$

Pentru situația 0 – 1 se ia o unitate de la poziția de ordin superior şi atunci 10 – 1 = 1.

Exemplul 1: x = 111101 şi y = 110100; x – y = ?

$$\begin{array}{r} 111101- \\ 110100 \\ \hline 1001 \end{array}$$

Exemplul 2: x = 1101000 şi y = 10011; x – y = ?

$$\begin{array}{r} \underline{7654321} \rightarrow \text{coloane} \\ 1101000- \\ 10011 \\ \hline 1010101 \end{array}$$

- prima coloană: avem 0 – 1 $\Rightarrow$ se ia o unitate de la poziția de ordin superioț $\Rightarrow$ 10 – 1 = **1**

- a doua coloană: trebuie să se scadă din 0 unitatea scăzută anterior şi unitatea lui y. De aceea, se ia o unitate de la poziția de ordin superior $\Rightarrow$10–1–1= 1 – 1 = **0**
- a treia coloană: trebuie să se scadă din 0 unitatea scăzută anterior. Astfel, se ia o unitate de la poziția de ordin superior $\Rightarrow$ 10 – 1 = **1**
- a patra coloană: se scade din 1 unitatea anterior scăzută$\Rightarrow$ 1–1 = **0**
- a cincea coloană: 0 – 1 $\Rightarrow$ se ia o unitate de la poziția de ordin superior $\Rightarrow$10 – 1 = **1**
- a şasea coloană: se scade din 1 unitatea anterior scăzută $\Rightarrow$1–1 = **0**
- a şaptea coloană: nu se scade nimiç $\Rightarrow$**1**

❒ Înmulțirea

Tabela înmulțirii în sistemul binar este:

$$\begin{cases} 0 \cdot 0 = 0 \\ 0 \cdot 1 = 0 \\ 1 \cdot 0 = 0 \\ 1 \cdot 1 = 1 \end{cases}$$

Exemplul 1: x = 10101 şi y = 1001; $x \cdot y = ?$

$$\begin{array}{r} 10101\cdot \\ 1001 \\ \hline 10101 \\ \underline{10101} \\ 10111101 \end{array}$$

Exemplul 2: x = 101101 şi y = 11101; $x \cdot y = ?$

```
      101101·
       11101
  ------------
      101101
     101101
    101101
   101101
  ----------
  10100011001
```

❒ **Împărțirea**

Fie x = 11011 şi y = 101. Atunci x : y se obține astfel:

```
11011 | 101
101   | 101
------|
==11  |
  101 |
------|
 =111 |
  101 |
------|
  =10 |
```

câtul este 101, iar restul 10.

1.4 Sistemul hexazecimal

Cifrele sistemului hexazecimal sunt: 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. Literele utilizate au următoarele interpretări în baza 10:

A = 10; B = 11; C = 12; D = 13; E = 14; F = 15

1.4.1 Conversia numerelor din sistemul hexazecimal în zecimal şi invers

Conversiile numerelor din hexazecimal în zecimal şi invers se efectuează conform celor prezentate în subcapitolele 1.2.1 şi 1.2.2.. De exemplu, conversia numărului $7A5_{16}$ în baza 10 se face astfel:

$$\begin{aligned} 7A5 &= 7\cdot 16^2 + A\cdot 16^1 + 5\cdot 16^0 \\ &= 7\cdot 256 + 10\cdot 16 + 5 \\ &= 1792 + 160 + 5 \\ &= 1957 \end{aligned} \quad , \qquad \text{deci } 7\text{A}5_{16} = 1957_{10}.$$

Pentru a afla corespondentul hexazecimal al unui număr în baza 10 se fac împărțiri succesive la 16, procedându-se ca la aflarea corespondentului binar.

$$\begin{aligned} 1937 &= 121 \cdot 16 + 1 \\ 121 &= 7 \cdot 16 + 9 \\ &= 0 \cdot 16 + 7 \end{aligned}$$

7 9 1

, deci $1937_{10} = 791_{16}$

1.4.2 Conversia numerelor din sistemul hexazecimal în binar şi invers

O împărțire la 16 este echivalentă cu patru împărțiri la 2. Orice cifră a sistemului cu baza 16 se scrie cu patru cifre binare. Utilizând acesta corespondență, numărul $4B0F13_{16}$ se scrie astfel în baza 2:

$$\underbrace{4}_{0100}\ \underbrace{B}_{1011}\ \underbrace{0}_{0000}\ \underbrace{F}_{1111}\ \underbrace{1}_{0001}\ \underbrace{3}_{0011}$$

Invers, numărul binar 11010110101011 are următorul corespondent în baza 16:

$$\underbrace{0011}_{3}\ \underbrace{0101}_{5}\ \underbrace{1010}_{A}\ \underbrace{1011}_{B}$$

1.4.3 Operații aritmetice în sistemul hexazecimal

În hexazecimal, operațiile de adunarea, respectiv scădere, se efectuează astfel:

❒ **Adunarea**

Modul de efectuare a operației de adunare este ilustrat în tabelul 1.1

Exemplul 1: x = 1AB02 şi y = A13; x + y = ?

```
  1
------
1AB02 +
  A13
------
1B515
```

- 3 + 2 = 5
- 1 + 0 = 1
- A + B = 15
- A + 1 = B

Exemplul 2: x = ABFF9 şi y = 9F38; x + y = ?

```
 1111
------
ABFF9 +
 9F38
------
B5F31
```

- 9 + 8 = 11
- F + 3 + 1 = 13
- F + F + 1 = 1F + 1 = 1F
- 9 + B + 1 = 15
- A + 1 = B

Tabelul 1.1. *Tabla adunării în sistemul hexazecimal*

+	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10
2	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11
3	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12
4	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13
5	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14
6	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15
7	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16
8	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17
9	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18
A	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19
B	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A
C	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B
D	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C
E	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D
F	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1F

❒ **Scăderea**

Exemplul 1: x = 1937A şi y = A65; x – y = ?

```
 1937A –
   A65
 ------
 18915
```

Exemplul 2: x = AB0003 şi y = BF21; x – y = ?

```
 AB0003 –
   BF21
 -------
 AA40E2
```

1.5 Reprezentarea internă a numerelor în calculator

1.5.1 Reprezentarea internă a numerelor întregi

Sistemul de numerație utilizat pentru reprezentarea datelor într-un sistem de calcul electronic este sistemul binar. Reprezentarea numerelor în acest sistem se poate face în mai multe forme, în funcție de mulțimea căreia îi aparține numărul. Operațiile (calculele) sunt efectuate fie de dispozitive aritmetice specializate pentru fiecare mod de reprezentare, fie prin succesiunea conversie-calcul-conversie, în cazul în care sistemul de calcul nu dispune decât de un singur dispozitiv specializat pentru calcul [2, 22].

Reprezentarea numerelor naturale, numită şi reprezentare aritmetică, se realizează pe lungimi standard, de 8, 16, 32, 64 biți. Dacă se cunoaşte numărul de biți pe care se reprezintă un număr natural, se poate stabili numărul maxim admis care poate fi reprezentat.

Acest număr maxim se obține prin ocuparea tuturor pozițiilor disponibile cu cifra 1.

Reprezentarea numerelor întregi este numită şi reprezentare algebrică. Ea este asemănătoare reprezentării aritmetice, cu deosebirea că prima poziție este ocupată de semnul numărului reprezentat. Prin convenție, dacă poziția de semn este 0, numărul este pozitiv, iar dacă este 1 numărul este negativ. De regulă, numerele întregi se reprezintă pe 8, 16, 32 şi 64 biți. Cunoscând numărul de poziții binare pe care se reprezintă un număr întreg, se poate determina intervalul admis pentru această reprezentare.

Reprezentarea numerelor negative se poate face în trei forme, astfel:

- în ***cod direct***, adică reprezentare prin mărime şi semn, care coincide cu reprezentarea numerelor întregi, având cifra 1 în poziția alocată semnului, urmată de cifrele binare ale numărului.
- în ***cod invers***, adică reprezentarea în complement față de 1. Fiecare cifră binară dintr-un număr reprezentat în cod invers este complementul față de 1.
- în ***cod complementar***, complementul față de 2, în care fiecare cifră binară îşi schimbă starea, adunând apoi 1 la cifra cea mai puțin semnificativă.

1.5.2 Reprezentarea numerelor reale în virgulă mobilă

Numerele reale se reprezintă într-un sistem de calcul sub formă fracționară, prin intermediul reprezentării în virgulă mobilă (flotantă). Forma generală de reprezentare a unui număr în virgulă mobilă este:

$$N = \pm 1, f \cdot B^{\pm E}$$

unde: N - este numărul reprezentat, $\pm$ - semnul (pozitiv sau negativ), f - partea fracționară, B - baza de numerație, E - exponentul (puterea).
În această formă, f se numeşte ***formă normalizată***. Reprezentarea numerelor reale în virgulă mobilă se poate face în două forme:

- virgulă mobilă simplă precizie (pe 32 biți);
- virgulă mobilă dublă precizie (pe 64 biți).

Forma de reprezentare în ***simplă precizie*** se prezintă astfel:

32	31 24	23 1
S	C = E + 127	M

unde:

S – bitul de semn care respectă aceeaşi condiție ca la numerele întregi: 0 pentru numerele pozitive, 1 pentru cele negative;
C – caracteristica;
E – exponentul;
M – mantisa (partea fracționară).

Pentru a nu se ocupa încă o poziție binară pentru semnul exponentului, s-a introdus noțiunea de *caracteristică*. Aceasta *este reprezentarea exponentului în exces de 127*. Rezultă în urma acestei convenții, că, dacă C > 127, exponentul E > 0, iar dacă C < 127, atunci E < 0.

Lungimea mantisei reprezintă de fapt precizia de reprezentare a numărului. Forma de reprezentare în ***dublă precizie*** foloseşte un număr de 52 de biţi pentru mantisă:

64	63 53	52 1
S	C = E + 127	M

unde:

S – bitul de semn;

C – caracteristica;

M – mantisa.

Atât la numerele în simplă precizie, cât şi la cele în dublă precizie, partea fracţionară este aliniată stânga, adică de la virgulă.

La formatul de reprezentare în virgulă mobilă, se observă că cifra 1 din faţa virgulei nu se reprezintă. Acest lucru se va avea în vedere, când se va interpreta numărul real exprimat în virgulă mobilă.

1.6 Codificarea datelor

Procedeul de reprezentare a informaţiei conform unui anumit format poartă denumirea de ***codificare.***

Necesitatea utilizării codurilor se impune pentru a asigura comunicaţia între utilizator şi sistemul de calcul, având în vedere că utilizatorului îi este specifică gândirea zecimală şi folosirea caracterelor alfabetului, în timp ce tehnica de calcul recunoaşte doar sistemul binar.

Se numeşte ***cod*** un set de simboluri elementare, împreună cu o serie de reguli potrivit cărora se formează aceste simboluri.

Operaţia de codificare este prezentă pe mai multe nivele de tratare a informaţiilor. Ea este prezentă pornind de la nivelul scăzut al sistemului de calcul (hardware, software de bază), până la nivelul codificării în ansamblul sistemului informaţional (coduri de materiale, coduri de produse, codul de bare etc.). Procedura de codificare, la nivelul sistemului informaţional, este în mare parte la alegerea utilizatorului. Cu cât se coboară spre nivelul elementar de prelucrare a informaţiei, cu atât sistemele de codificare sunt mai rigide, ele fiind standardizate din considerente de compatibilitate în utilizarea tehnicii de calcul.

Codurile alfanumerice sunt coduri binare utilizate pentru reprezentarea caracterelor alfanumeric.

Prin *caractere alfanumerice* se înţelege:

- literele mari şi mici ale alfabetului;
- cifrele de la 0 la 9;
- semnele de punctuaţie;
- operatorii aritmetici şi de relaţie;
- alte caractere speciale.

În general, prin caractere alfanumerice se înţeleg toate caracterele care pot fi introduse de la tastatura unui calculator. Numărul acestor caractere este mai mare decât 64 şi, din acest motiv, pentru reprezentarea binară a caracterelor alfanumerice sunt necesare cel puţin 7 poziţii binare. Cum numărul de caractere de codificat este limitat şi relativ redus, funcţia de codificare alfa-numerică se defineşte tabelar.

Cele mai cunoscute coduri alfanumerice sunt::

- Codul Extended Binary Coded decimal Interchange Code **(EBCDIC)** este un cod alfanumeric pe 8 biţi, proiectat pentru codificarea informaţiilor, la calculatoarele din seria IBM 360; Codul a fost utilizat până prin anii ’70.
- Codul American Standard Code for Information Interchange (**ASCII)** reprezintă un standard actual de codificare alfanumerică pe 7 biţi; Este un cod alfanumeric, cu recomandare International Standard Organization (ISO) şi este utilizat de multe tipuri de calculatoare şi de majoritatea echipamentelor periferice care lucrează cu aceste calculatoare.
- Codul **UNICODE**. Este un cod alfanumeric pe 16 biţi care îşi propune să definească un cod standard pentru simbolurile alfanumerice specifice tuturor limbilor de pe planetă.

Prin dezvoltarea sistemelor de coduri pe 7 şi 8 biţi, aşa-numitele coduri pe octet, către coduri pe doi octeţi sau coduri Double-Bit Code System (DBCS) s-a creat o familie de coduri care se doreşte a fi unificată de sistemul de codificare UNICODE. Înlocuirea codului ASCII cu UNICODE ar avea avantajul unificării sistemului de codificare a caracterelor folosite oriunde în lume.

În acelaşi timp se evidenţiază şi unele dezavantaje:

- utilizând UNICODE pe 16 biţi, dimensiunea fişierelor text se dublează, în comparaţie cu cea a fişierelor codificate ASCII;
- programele care utilizează codul ASCII vor trebui adaptate pentru a putea recunoaşte codul UNICODE.