Introducere

Microprocesorul este o componentă care a revoluționat informatica. El are o funcționare transparentă (invizibilă) pentru utilizatorul unui sistem de calcul, deoarece între microprocesor şi utilizator se află de regulă un program special - sistemul de operare. Aplicațiile microprocesoarelor au atins practic toate domeniile: ştiințific, tehnic, educațional, economic, financiar, social - politic şi cultural.

Microprocesorul a apărut în 1971, istoria sa fiind tipic americană. În 1968, doi cercetători, Robert N. Noyce şi Gordon E. Moore, părăsesc societatea *Fairchild* pentru a fonda *Intel*. Ambiția lor era de a exploata tehnologia LSI (*Large Scale Integration*) ce se năştea atunci. Primele produse au fost circuite de memorie cu semiconductoare (inelele de ferită dominau epoca). În 1969 primul circuit realizat a fost memoria bipolară Schottky 3101, de 64 biți şi în acelaşi an, memoria MOS 1101, de tip static de 256 biți. Apoi în 1970 realizează memoria dinamică MOS 3101 de 1024 biți cu timp de acces de 300 ns.

În 1969, o societate japoneză, astăzi dispărută, Busicom, cere firmei Intel realizarea în producție de serie a unei familii de 5 circuite integrate pentru calculatoare de birou performante. Inginerul Ted Hoff de la Intel a schimbat proiectul. El a considerat prea complex şi costisitor proiectul inițial al firmei Busicom şi a redus la patru numărul circuitelor prin reducerea complexității instrucțiunilor şi introducerea unei memorii complementare. Cele patru circuite erau: unitatea centrală (CPU), o memorie ROM pentru programe de aplicație, o memorie RAM pentru date şi un registru specializat în intrări-ieşiri (I/O). Unitatea centrală s-a numit *Intel 4004* - primul microprocesor din lume. Cu cele 2300 tranzistoare MOS şi 60 000 operații pe secundă, Intel 4004 avea aproximativ aceeaşi putere de calcul ca şi calculatorul ENIAC - considerat a fi la originea informaticii.

Firma Busicom a fost convinsă cu greu să accepte noua formulă, dar în final a acceptat şi în 1970 a fabricat şi comercializat în jur de 100 000 de calculatoare.

Apariția primului microprocesor nu a constituit un eveniment deosebit în lumea informatică. El aplica tehnologiile noi de integrare - foarte cunoscute - pentru a realiza pe un singur "*chip*" schema de principiu a unui calculator, care era de asemenea cunoscută.

Chiar firma Intel îşi punea problema reutilizării noii serii de 4 circuite integrate; ideea a fost să utilizeze această serie ca "locomotivă" pentru a vinde memorii. Intel a cumpărat dreptul de comercializare a seriei

dar nu cu mare succes: prețul de 30 - 40 $ pentru câteva circuite a fost considerat de clienți prea ridicat.

În 1972 Intel lansează versiunea de 8 biți a procesorului, Intel 8008 care este bine primită pe piață. Procesarea datelor de 8 biți era foarte convenabilă pentru reprezentarea caracterelor alfanumerice.

Dezvoltând conceptul de bază, Intel propune în 1974 microprocesorul 8080, care devine cu adevărat primul microprocesor de uz general din lume. De 10 ori mai performant decât 8008, el execută 290.000 operații pe secundă, poate adresa 64 ko de memorie, dispune de densitate de integrare sporită. Este comercializat la prețul de 360 $ în aprilie 1974.

Succesul de piață al lui Intel 8080 a trezit concurența: Motorola prezintă concurentul său M 6800 urmat de Z80 al firmei Zilog şi de produsul firmei MOS Technology, 6502 (unitatea centrală a lui Apple 2 şi a primelor microcalculatoare Commodore). Răspunsul din partea lui Intel a fost procesorul 8085 care adaugă lui 8080 caracteristici foarte interesante şi încă un avantaj important - o singură tensiune de alimentare de 5 V (8080 necesită trei tensiuni de alimentare).

În 1976, Intel, care avea un proiect de microprocesor pe 32 biți (Intel 432), renunță la acesta şi dezvoltă varianta pe 16 biți, 8086. Este o extensie a arhitecturii devenite clasice a lui 8080. Programul de fabricație al lui 8086 este condus de un francez stabilit în Statele Unite, Jean - Claud Cornet. Desenul lui 8086, pe hârtie, ocupa o suprafață de 10 metri pe 10 metri (100 m.p.). Este comercializat în 1978.

În 1979, Motorola prezintă procesorul său de 16 biți, M 68000 .

Pentru Intel, istoria continuă cu 8088, o variantă de 8 biți a lui 8086, ce va fi adoptată de IBM pentru primul său microcalculator PC-XT, cu 80186/188 ale cărui aplicații au rămas misterioase, cu 80286 adoptat de IBM pentru microcalculatoarele PC-AT, apoi 386, 486 şi Pentium.

Microprocesoarele au devenit nu numai o industrie completă prin ele însele dar au determinat o creştere spectaculoasă a cererii de circuite integrate diverse şi în special de memorii. Ele sunt utilizate nu numai ca unități centrale în microcalculatoare dar şi în calculatoarele mari, în care numeroase unități care funcționează în paralel, conțin microprocesoare. Dezvoltarea spectaculoasă a microprocesoarelor a produs o enormă piață de periferice, de programe (*Software*), de servicii etc.

Impactul microprocesoarelor poate fi comparat cu cel al automobilelor. De exemplu, în 1995 s-au vândut în toată lumea circa 50 milioane de microcalculatoare contra 40 milioane de automobile. S-a propus ca numărul de microprocesoare utilizate (pe locuitor) să constituie un indicator al progresului tehnic pentru țările industrializate.

1 Reprezentarea informației în sistemele numerice

Studiul microprocesoarelor impune referirea frecventă la sistemele de numerație ca urmare a reprezentării informației. Sistemul de numerație uzual este cel zecimal, dar circuitele electronice numerice lucrează în cel binar, care utilizează doar două simboluri pentru reprezentarea informației, *0* şi *1*.

Sistemul binar fiind incomod pentru utilizatorul uman, se utilizează ca intermediar sistemul hexazecimal, cu 16 simboluri, conversiile binar-hexa şi hexa-binar fiind foarte simple.

Definiții

Referitor la reprezentarea binară, singura posibilă în sistemele actuale cu microprocesor, se utilizează următoarele noțiuni:

-bit (**Bi**nary Digi**t**, pe scurt **b**) pentru o cifră binară **0** sau **1**;

-nibble (pe scurt **n**) pentru un şir de 4 biți;

-byte sau **octet** (pe scurt **B**) pentru un şir de 8 biți;

-word sau **cuvânt** (pe scurt **w**) pentru un şir de 16 biți;

-double word sau **dublu cuvânt** (pe scurt **dw**) pentru 32 biți;

În binar prefixul kilo este asociat cu $1024 = 2^{10}$; astfel:

$1k = 1024 \cong 10^3$; $1Mega = 1024 \times 1024 \cong 10^6$;

$1Giga = 2^{30} \cong 10^9$; $1Tera = 2^{40} \cong 10^{12}$;

1.1 Sisteme de numerație

Un sistem de numerație este o mulțime finită de simboluri - numite cifre împreună cu un set de reguli pentru reprezentarea numerelor; numărul simbolurilor reprezintă baza sistemului de numerație.

De exemplu, Sistemul Roman are şapte simboluri: I, V, X, L, C, D, M. Fiecare cifră are o valoare numerică (pondere) 1, 5, 10, 50, 100, 500, 1000.

- numărul reprezentat se obține prin însumarea ponderilor, dacă sunt descrescătoare: MDLXX = 1000+500+50+10+10=1570;

- dacă un simbol este urmat de altul, cu pondere mai mare, valoarea numerică se obține prin diferență: MCM = 1000+(1000-100)=1900;

- un simbol se poate repeta consecutiv de cel mult 3 ori (cu excepția simbolului M); astfel 40 se scrie corect XL şi nu XXXX;

MDXXX=1530; MCMXCIX=1999; MMII = 2002;

Sistemul Roman este utilizat în prezent doar pentru numerale ordinale.

Sistemele uzuale sunt **poziționale**, adică ponderea unei cifre depinde de locul pe care aceasta în ocupă în şirul de cifre al reprezentării.

Fie un sistem de numerație pozițional $\Re$ şi mulțimea de simboluri

$$S = \{c_1, c_2, \ldots, c_B\} ;$$

un număr *N (întreg)* se scrie în acest sistem astfel:

$$N = c_{i_1} c_{i_2} \ldots c_{i_n} = \sum_{m=1}^{n} c_{i_m} \cdot B^{n-m} = c_{i_1} \cdot B^{n-1} + c_{i_2} \cdot B^{n-2} + \ldots + c_{i_n} \cdot B^0 \quad ,$$

unde *B* este baza sistemului de numerație (numărul de simboluri) iar indicii sunt în domeniul 1, 2, 3, . . . ,*B*.

Exemple:

1. În sistemul zecimal (baza *B* = 10), mulțimea simbolurilor este
   S = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
   numărul 1987 se scrie: $1987 = \mathbf{1} \cdot 10^3 + \mathbf{9} \cdot 10^2 + \mathbf{8} \cdot 10^1 + \mathbf{7} \cdot 10^0$
2. În sistemul binar (baza *B* = 2), mulțimea simbolurilor este S = {0, 1};
   numărul 21 se scrie $10101 = \mathbf{1} \cdot 2^4 + \mathbf{0} \cdot 2^3 + \mathbf{1} \cdot 2^2 + \mathbf{0} \cdot 2^1 + 1 \cdot 2^0 =$
   16 + 0 + 4 + 0 + 1 = 21.
3. În sistemul hexazecimal (baza *B* = 16), mulțimea simbolurilor este
   S = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F};
   numărul 1000 se scrie $3E8 = 3 \cdot 16^2 + E \cdot 16^1 + 8 \cdot 16^0 =$
   768 + 224 + 8 = 1000.

1.1.1. Conversia unui număr dintr-o bază în alta

Este necesar adeseori ca un număr dat într-o bază de numerație să fie exprimat în altă bază de numerație.

1.Conversia **din baza 10 într-o bază *B*** se poate face prin exprimarea numărului sub formă de polinom în "variabila" *B:*

$N = a_0 B^n + a_1 B^{n-1} + \ldots + a_{n-1} B + a_n$.

În această exprimare a numărului *N*, coeficienții $a_0, a_1, \ldots, a_n$ sunt chiar cifrele reprezentării lui *N* în baza *B,* în această ordine.

Rezultă următorul algoritm de conversie:

1. Se calculează a_n, restul împărțirii lui *N* prin *B*: $N = Q \cdot B + a_n$;

$N = (a_0 B^{n-1} + a_1 B^{n-2} + \ldots + a_{n-1}) \cdot B + a_n$;

2.Se calculează a_{n-1} , restul împărțirii lui *Q* prin *B*: $Q = Q_1 \cdot B + a_{n-1}$:

3. Se repetă pasul 2 cu fiecare cât, până când se obține câtul = 0.

Tabelul 1

Zecimal (Z)	Binar (B)	ZCB (DCB)	Octal (Q)	Hexazecimal (H)

0	0	0 0 0 0	0	0
1	1	0 0 0 1	1	1
2	1 0	0 0 1 0	2	2
3	1 1	0 0 1 1	3	3
4	1 0 0	0 1 0 0	4	4
5	1 0 1	0 1 0 1	5	5
6	1 1 0	0 1 1 0	6	6
7	1 1 1	0 1 1 1	7	7
8	1 0 0 0	1 0 0 0	10	8
9	1 0 0 1	1 0 0 1	11	9
10	1 0 1 0		12	A
11	1 0 1 1		13	B
12	1 1 0 0		14	C
13	1 1 0 1		15	D
14	1 1 1 0		16	E
15	1 1 1 1		17	F
16	1 0 0 0 0		20	10

4. Resturile obținute, scrise de la stânga la dreapta începând cu ultimul rest este reprezentarea numărului *N* în baza *B*.

Exemplu: N = 37; B = 2;

se scrie succesiv: 37 = 18 x 2 + **1** ;

18 = 9 x 2 + **0** ;

9 = 4 x 2 + **1**;

4 = 2 x 2 + **0** ;

2 = 1 x 2 + **0** ;

1 = 0 x 2 + **1** ;

punând alături resturile în ordinea inversă obținerii lor, se obține: 100101;

$37_{10} = \mathbf{1} \cdot 2^5 + \mathbf{0} \cdot 2^4 + \mathbf{0} \cdot 2^3 + \mathbf{1} \cdot 2^2 + \mathbf{0} \cdot 2^1 + \mathbf{1} \cdot 2^0 = 100101_2$

În mod analog se face conversia din baza 10 în orice bază; de exemplu pentru conversia numărului 37 în hexazecimal, se scrie succesiv:

37 = 2 x 16 + **5**; 2 = 0 x 16 + **2**; se obține:

$37_{10} = \mathbf{2} \cdot 16^1 + \mathbf{5} \cdot 16^0 = 25_{16}$.

În tabelul 1.1 sunt prezentate comparativ primele 16 numere naturale în cele mai utilizate sisteme de numerație: zecimal, binar, octal şi hexazecimal. Se poate observa în mod direct codificarea binară a celor 16 cifre hexazecimale şi de asemenea relația dintre sistemul octal şi binar.

2. Pentru conversia din **hexazecimal în binar**, fiecare cifră (hexa) se înlocuieşte prin codul său binar (4 cifre binare) conform tabelului 1.

Exemplu: 1 A B F 2 (H) = 0001 1010 1011 1111 0010 unde primele trei zerouri pot fi omise.

3.Pentru conversia din **binar în hexazecimal**, se separă reprezentarea binară în grupe de patru cifre de la dreapta spre stânga şi apoi fiecare grup se înlocuieşte cu cifra hexazecimală corespunzătoare din tabelul 1.

Exemplu: 10001011001010 = 10 0010 1100 1010 = 2 2 C A (H)

4. Pentru **conversia dintr-o bază *B* în baza 10** se utilizează formula de definiție a numărului în baza *B:*

$$N = a_0 B^n + a_1 B^{n-1} + \ldots + a_{n-1} B + a_n \quad ,$$

unde coeficienții sunt cifrele reprezentării în baza *B*.

De exemplu, $678_9 = 6 \cdot 9^2 + 7 \cdot 9^1 + 8 \cdot 9^0 = 486 + 63 + 8 = 557_{10}$.

1.1.2. Reprezentarea numerelor fracționare

În acest caz reprezentarea are două părți: partea întreagă şi partea fracționară:

$$N = a_0 B^n + a_1 B^{n-1} + \ldots + a_{n-1} B + a_n B^0 + a_{n+1} B^{-1} + a_{n+2} B^{-2} + \ldots + a_{n+m} B^{-m} \quad ,$$

unde puterile negative ale bazei marchează partea frațională.

Astfel, numărul binar 101,111 se scrie:

$101,111 = 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 + 1 \cdot 2^{-1} + 1 \cdot 2^{-2} + 1 \cdot 2^{-3} = 5 + 0,5 + 0,25 + 0,125;$

aşadar,

$$101,111_2 = 5,875_{10}.$$

Conversia unui număr fracționar subunitar din baza 10 în binar, se face prin multiplicări succesive cu 2 şi reținerea părții întregi:

$0,8125_{10} = (\ ?\ )_2$;

0,8125x 2 = **1**,6250; 0,625 x 2 = **1**,250 ; 0,25 x 2 = **0**,50 ; 0,5 x 2 = **1**,0 ;

$0,8125_{10} = 0,1101_2$.

1.1.3. Reprezentarea numerelor pozitive şi negative

În sistemul binar, prima şi ultima cifră binară (bit) dintr-un şir au denumiri specifice:

- primul bit (din extrema stângă) este numit "cel mai semnificativ bit" sau **MSB** (*Most Significant Bit*);

- ultimul bit (din extrema dreaptă) este numit " cel mai puțin semnificativ bit" sau **LSB** (*Least Significant Bit*).

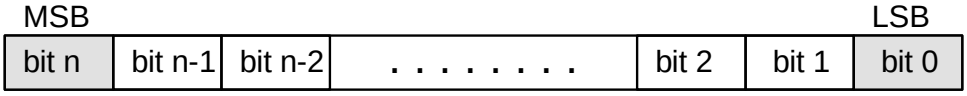

Fig. 1

În sistemul binar existând doar două simboluri, pentru specificarea semnului s-a introdus următoarea convenție: pentru "+" se utilizează "0" iar pentru "-" se utilizează "1"; bitul de semn este MSB.

Exemplu: numărul negativ -15 se reprezintă: **1 0 0 0 1 1 1 1**
ceea ce se mai numeşte "octet cu semn"; în timp ce valoarea unui octet fără semn este cuprinsă între 0 . . 255, cea a unui octet cu semn are domeniul -127. . +127.

Numerele negative se pot reprezenta în trei moduri: ca număr binar cu semn (exemplul de mai sus), în complement față de 1 şi în complement față de 2; ultimele două s-au introdus pentru înlocuirea scăderii cu adunarea.

Complementul față de *1* al unui număr binar se obține prin schimbarea fiecărui bit în bitul complementar: *1* devine *0* iar *0* devine *1*.

Exemplu: 0100 1011 în compl. 1 devine 1011 0100.

Complementul față de *2* al unui număr binar se obține din complementul față de *1* prin adunare cu unu. Este utilizat pentru înlocuirea unei scăderi, cu adunarea complementului față de 2.

Exemplu:

0100 1011 în compl. 2 devine 1011 0100 + 1 = 1011 0101.

Să verificăm înlocuirea scăderii cu adunarea complementului:

Fie N = 0101 0000_2 = 80_{10} şi M = 0100 1011_2 = 75_{10}; atunci avem:

N - M = 0101 0000 - 0100 1011 = 0101 0000 + (80)
1011 0101 (compl.2) (-75)
0000 0101 = 5_{10} (5)

Adunarea şi scăderea vor da rezultate corecte dacă nu apar depăşiri ale limitelor de reprezentare (*overflow* - depăşirea numărului de biți ai reprezentării sau *borrow* - împrumut la rangul superior lui MSB).

În cazul înmulțirii a două numere în binar se consideră că rezultatul va avea o lungime dublă (număr de biți) față de lungimea operanzilor.

În cazul împărțirii se detectează tentativa de împărțire prin zero care poate produce rezultate eronate.

Toate microprocesoarele au în setul de instrucțiuni operațiile de adunare şi scădere dar nu toate au operațiile de înmulțire şi împărțire. Acestea pot fi implementate în programe utilizând diferiți algoritmi.

1.1.4. Reprezentarea numerelor în virgulă fixă şi în virgulă mobilă

Reprezentarea ***în virgulă fixă*** utilizează un număr fix de biți pentru partea întreagă şi respectiv pentru cea fracționară. Pentru partea întreagă se utilizează regula de reprezentarea a numerelor întregi cu semn iar pentru cea fracționară, regula de reprezentare a întregilor fără semn. Se utilizează, de regulă, următoarele convenții:

- se rezervă un şir de biți pentru numărul total de cifre ale numărului (inclusiv bitul de semn) - zona I;
- se rezervă un şir de biți pentru exprimarea numărului de cifre ale părții fracționare - zona II;
- se reprezintă numărul în binar prin şirul de biți care rezultă alăturând zona I, zona II, partea întreagă, partea fracționară, fără nici o delimitare.

Reprezentarea în virgulă fixă se utilizează în unele sisteme de conducere cu calculator a maşinilor unelte (sisteme de poziționare) şi în domeniul financiar - contabil.

La acestea din urmă, reprezentarea în virgulă fixă are avantajul că toate puterile lui 10 care se încadrează în domeniul de valori stabilit, sunt reprezentate exact. Pentru a reprezenta sume de bani, sunt suficiente două cifre după virgulă. La un bilanț financiar, este esențial ca operațiile aritmetice să se facă exact. Din aceste motive, în programele aplicative din domeniul economic se utilizează reprezentarea numerelor fracționare în virgulă fixă.

Reprezentarea ***în virgulă mobilă*** are un format standard în funcție de numărul de biți ai reprezentării, fiind utilizate reprezentările pe 32 de biți, 64 de biți şi 80 de biți.

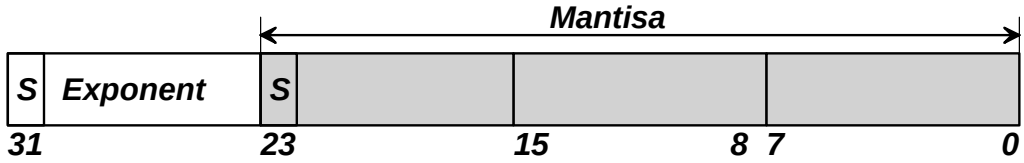

Fig. 2 Reprezentarea în virgulă mobilă pe 32 de biți.

Pentru reprezentare, numărul se exprimă ca produs dintre un număr subunitar cu semn (mantisa) şi o putere a lui 2 (exponent = întreg cu semn):

$$N = M \cdot 2^{Exp}, \quad 2^{-1} \le M < 2^{0};$$

Mantisa M se numeşte *normalizată* dacă prima cifră după virgulă este diferită de zero.

Din figura 2 rezultă că în reprezentarea în virgulă mobilă pe 32 de biți se alocă 8 biți pentru exponent şi 24 de biți pentru mantisă; în acest caz, domeniul de reprezentare este $\pm(2^{-127}\ldots 2^{127})$, fiind mult extins față de reprezentarea în virgulă fixă pe acelaşi număr de biți.

Exponentul se reprezintă de obicei deplasat, în sensul că se memorează un număr de forma E + K, unde K este o constantă astfel aleasă încât E + K să fie totdeauna pozitiv. În acest mod nu mai este necesar bitul de semn, care este totdeauna zero.

Prima cifră semnificativă a mantisei normalizate fiind 1, această cifră nu se mai reprezintă, câştigâdu-se astfel un bit în spațiul de memorare (în care se poate reprezenta bitul de semn).

Cea mai utilizată reprezentare standardizată este cea pe 32 de biți numită în **simplă precizie**.

Exponentul are 8 biți şi se reprezintă intern adunat cu 127. Mantisa este normalizată şi înmulțită cu 2, astfel încât condiția de normalizare devine: $1 \le M < 2$.

Bitul cel mai reprezentativ al mantisei nu se mai reprezintă intern, fiind totdeauna 1. Formula de reprezentare este dată de relația:

$$x = (-1)^s \cdot 1 \cdot f_{22}\ f_{21} \ldots f_1\ f_0 \cdot 2^{e-127} \quad ,$$

unde ***s*** este bitul de semn, ***e*** este exponentul mărit cu 127 iar cei 23 de biți formează mantisa normalizată, notată de obicei cu ***f*** (de la flotant); cel de-al 24 -lea bit al mantisei, de valoare 1, nu se reprezintă intern.

În tabloul următor sunt prezentați cei 4 octeți în ordinea aşezării lor în memorie, în sensul crescător al adreselor.

Tab.2

f7	**f6**	**f5**	**f4**	**f3**	**f2**	**f1**	**f0**
f15	**f14**	**f13**	**f12**	**f11**	**f10**	**f9**	**f8**
e0	**f22**	**f21**	**f20**	**f19**	**f18**	**f17**	**f16**
s	**e7**	**e6**	**e5**	**e4**	**e3**	**e2**	**e1**

Exponentul deplasat (mărit cu 127) ia valori între 1 şi 254. Valoarea 0 este permisă numai pentru reprezentarea valorii reale 0.0 (când s=0, f=0) iar valoarea 255 (0FFH) se utilizează pentru cazuri de excepție, -INF şi +INF care sunt considerate depăşiri aritmetice.

- INF : e = 0FFH f = 7FFFFFH s = 1

+INF : e = 0FFH f = 7FFFFFH s = 0

Cel mai mic număr pozitiv reprezentabil este $1.17 \cdot 10^{-38}$, caracterizat de s=0, f=0, e=1.

Cel mai mare număr pozitiv reprezentabil are valoarea aproximativă $3.4 \cdot 10^{38}$, caracterizat de s=0, f=7FFFFFH, e=0FEH.

Tab 3.:

Număr	**Reprezentare pe 32 de biți**	**Obs.**
0,0	**00 00 00 00**	s=0, f=0, e=0

1,0	**00 00 80 3F**	s=0, f=0, e=7F H
2,0	**00 00 00 40**	s=0, f=0, e=80 H
0,5	**00 00 00 3F**	s=0, f=0, e=7E H
-1,0	**00 00 80 BF**	s=1, f=0, e=7F H

Observaţii:

1. La aceeaşi mantisă putem avea exponenţi diferiţi, ceea ce înseamnă că virgula nu este fixă ci se deplasează în funcţie de mărimea şi semnul exponentului. De aceea numele reprezentării este "în virgulă mobilă" sau "în virgulă flotantă". Avantajul reprezentării este domeniul de valori foarte extins pentru aplicaţii obişnuite: $-10^{38} \div 10^{38}$.

2. Se reprezintă exact numai numerele reale care se pot exprima ca sume de puteri pozitive şi negative ale lui 2; celelalte numere, ca de exemplu 0.3, 0.7, 0.001 etc., nu pot fi reprezentate exact în virgulă mobilă. De aceea, acest sistem de reprezentare nu se utilizează în programele de aplicaţii pentru domeniile economico - financiare.

3. Calculele aritmetice sunt aproximative dar precizia este suficient de bună pentru aplicaţiile uzuale. În cazul reprezentării în simplă precizie, pe 32 de biţi, eroarea de reprezentare este de $\pm 10^{-7}$.

În cazul reprezentării în ***dublă precizie***, pe 64 de biţi, mantisa este de 53 de biţi iar exponentul de 11 biţi (deplasat cu 1023).

Valoarea unui număr real în dublă precizie este dată de expresia:

$$x = (-1)^s \cdot 1 \cdot f_{51}\, f_{50}\, f_{49} \ldots f_1\, f_0 \cdot 2^{e-1023} \quad ,$$

unde ***s*** este bitul de semn, ***e*** este exponentul mărit cu 1023 iar cei 52 de biţi formează mantisa normalizată, notată cu ***f*** ; cel de-al 53 -lea bit al mantisei, de valoare 1, nu se reprezintă intern.

Structura celor 8 octeţi, aşa cum sunt stocaţi în memorie, în ordinea crescătoare a adreselor este dată în tabloul următor (4).

Tab. 4

f7	**f6**	**f5**	**f4**	**f3**	**f2**	**f1**	**f0**
f15	**f14**	**f13**	**f12**	**f11**	**f10**	**f9**	**f8**
f23	**f22**	**f21**	**f20**	**f19**	**f18**	**f17**	**f16**
f31	**f30**	**f29**	**f28**	**f27**	**f26**	**f25**	**f24**
f39	**f38**	**f37**	**f36**	**f35**	**f34**	**f33**	**f32**
f47	**f46**	**f45**	**f44**	**f43**	**f42**	**f41**	**f40**
e3	**e2**	**e1**	**e0**	**f51**	**f50**	**f49**	**f48**
s	**e10**	**e9**	**e8**	**e7**	**e6**	**e5**	**e4**

Reprezentările de 32 şi 64 biţi sunt formate normalizate recunoscute de forurile ştiinţifice internaţionale.

Reprezentarea ***în precizie extinsă***, pe 80 de biți, are o mantisă normalizată de 64 de biți şi un exponent de 15 biți (deplasat cu +16383).

Valoarea unui număr real în dublă precizie este dată de expresia:

$$x = (-1)^s \cdot 1 \cdot f_{62}\, f_{61}\, f_{60} \ldots f_1\, f_0 \cdot 2^{e-16383} \quad ,$$

unde ***s*** este bitul de semn, ***e*** este exponentul mărit cu 16383 iar cei 63 de biți formează mantisa normalizată, notată cu ***f*** ; cel de-al 64 -lea bit al mantisei, de valoare 1, se reprezintă intern ca bit constant de valoare 1 (cu excepția numărului 0.0 când şi acest bit are valoarea 0.

Structura celor 10 octeți, aşa cum sunt stocați în memorie, în ordinea crescătoare a adreselor este dată în tabloul următor (5).

Tab.5

f7	**f6**	**f5**	**f4**	**f3**	**f2**	**f1**	**f0**
f15	**f14**	**f13**	**f12**	**f11**	**f10**	**f9**	**f8**
f23	**f22**	**f21**	**f20**	**f19**	**f18**	**f17**	**f16**
f31	**f30**	**f29**	**f28**	**f27**	**f26**	**f25**	**f24**
f39	**f38**	**f37**	**f36**	**f35**	**f34**	**f33**	**f32**
f47	**f46**	**f45**	**f44**	**f43**	**f42**	**f41**	**f40**
f55	**f54**	**f53**	**f52**	**f51**	**f50**	**f49**	**f48**
1	**f62**	**f61**	**f60**	**f59**	**f58**	**f57**	**f56**
e7	**e6**	**e5**	**e4**	**e3**	**e2**	**e1**	**e0**
s	**e14**	**e13**	**e12**	**e11**	**e10**	**e9**	**e8**

Familia de microprocesoare Intel 80x86 dispune de circuite specializate (8087, 80287, 80387- coprocesoare aritmetice) pentru calcule în virgulă mobilă, care utilizează toate cele trei forme de reprezentare.

Intel 486 şi Pentium conțin coprocesoare aritmetice în structura lor internă. Limbajul de asamblare specific acestor procesoare dispune de directive pentru definirea constantelor şi variabilelor în toate cele trei forme de reprezentare.

1.1.5. Coduri binare

Sistemul binar prezentat anterior este numit adesea *binar natural*. La începuturile informaticii toate programele şi datele erau transformate în cod binar de către utilizator şi introduse în calculator sub această formă. Şirurile nesfârşite de 0 şi 1 în binar natural erau dificil de introdus, de verificat, de corectat. Erorile erau inevitabile.

Pentru reducerea acestor dificultăți, au fost inventate variante ale codului binar mai uşor de folosit şi mai rezistente la erori. Un asemenea cod

este cel Zecimal Codat Binar sau *BCD* în engleză (*Binary Coded Decimal*). Prin utilizarea codului BCD o mare parte din problemele de conversie sunt transferate calculatorului. Operațiile aritmetice efectuate în cod BCD necesită corecții ale rezultatului (există instrucțiuni de corecție) pentru ca acesta să corespundă codului binar natural. În esență, regulile sunt:

- fiecare cifră zecimală este înlocuită de 4 biți = 1 *nibble*;
- se utilizează cifrele zecimale 0, 1, 2, . . ,9;
- valorile binare superioare lui 9 = 1001, sunt interzise.

Se utilizează două forme ale codului BCD:

Tab. 6

Zecimal (Z)	**Binar (B)**	**BCD compactat**	**BCD necompactat**
0	0	0 0 0 0	0 0 0 0 0 0 0 0
1	1	0 0 0 1	0 0 0 0 0 0 0 1
2	1 0	0 0 1 0	0 0 0 0 0 0 1 0
3	1 1	0 0 1 1	0 0 0 0 0 0 1 1
4	1 0 0	0 1 0 0	0 0 0 0 0 1 0 0
5	1 0 1	0 1 0 1	0 0 0 0 0 1 0 1
6	1 1 0	0 1 1 0	0 0 0 0 0 1 1 0
7	1 1 1	0 1 1 1	0 0 0 0 0 1 1 1
8	1 0 0 0	1 0 0 0	0 0 0 0 1 0 0 0
9	1 0 0 1	1 0 0 1	0 0 0 0 1 0 0 1

1. *BCD compactat*: 4 biți pentru o cifră zecimală = digit (4 biți / digit);
2. *BCD extins* sau necompactat: un octet pentru un digit (8 biți / digit); în acest caz primii 4 biți ai octetului nu sunt utilizați (rămân 0).

Tab.7

Zecimal (Z)	Binar (B)	BCD (compactatat) zeci unități	BCD (necompactat) zeci unități
15	1111	0001 0101	000000001 00000101
31	100001	0011 0001	000000011 00000001
70	1000110	0111 0000	000000111 00000000
81	1010001	1000 0001	000001000 00000001
99	1100011	1001 1001	000001001 00001001
100	1100100	0001 0000 0000	00000001 00000000 00000000

1.1.6. Erori în BCD şi corecții

În BCD compactat există 6 combinații de patru biți inexistente şi anume cele corespunzătoare numerelor 10, 11, 12, 13, 14, 15, respectiv 1010, 1011, 1100, 1101, 1110, 1111. În cazul operațiilor cu numere în BCD apar erori care trebuie corectate automat de către microprocesor.

Microprocesoarele dispun de instrucțiuni specifice care corectează rezultatul în funcție de tipul operației, dar acestea sunt introduse în programe de către programator.

1. Depăşirea capacității digitului inferior:

Iată un prim caz de eroare. Să presupunem că adunăm în BCD 5+5; rezultatul nu mai poate fi reprezentat în BCD pe un singur digit (cel inferior), dar microprocesorul lucrează de fapt numai în binar natural şi rezultatul va fi:

0000 0101+
0000 0101
0000 1010 (codul 1010 nu există în BCD)

Principiul corecției este simplu: dacă digitul inferior are o combinație inexistentă, se adună această combinație binară cu 0110 = 6 în zecimal, ceea ce corespunde numărului de combinații binare interzise:

0000 1010+
0000 0110
0001 0000 = 10 (5+5=10), deci corect.

2. Depăşirea capacității digitului superior.

Problema este aceeaşi dar este afectat digitul superior. În acest caz corecția se face adunând 60 scris în BCD : 0110 0000.

Calculele se desfăşoară astfel:

	0101 0000 +	(50 în BCD)
	0101 0000	(50 în BCD)
	1010 0000 +	(cod inexistent în BCD)
Corecție:	0110 0000	(se adună 60)
	1 0000 0000	(se obține 100 în BCD, corect)

Se observă că s-a generat transport spre stânga care este memorat de un indicator de transport.

3.Generarea unui transport între cei doi digiți

În acest caz eroarea este produsă prin propagarea unui bit de transport de la digitul inferior spre cel superior. Acest tip de eroare apare, de exemplu, la adunarea 8+9=17, în BCD. Corecția se face, din nou, prin adăugarea numărului 06 în BCD:

	0000 1000+	(8 în BCD)
	0000 1001	(9 în BCD)
	0001 0001 +	(11 în BCD - incorect)
Corecție:	0000 0110	(se adună 06 în BCD)
	0001 0111	(rezultat corect: 17! în BCD)

În unitățile aritmetice ale microprocesoarelor, transportul de la bitul 4 spre bitul 5 (între cei doi digiți) este memorat de un indicator special numit "de semitransport " iar valoarea sa determină operația de corecție.

4. Transport generat de digitul superior

Dacă digitul superior al sumei generează transport, corecția se face tot prin adunarea cu 0110 la digitul superior:

	1000 0000 +	(80 în BCD)
	1001 0000	(90 în BCD)
	1 0001 0000	(110 în BCD, rezultat greşit)
Corecție:	0110 0000	(se adună 60 în BCD)
	1 0111 0000	(170 în BCD, rezultat corect)

5. Combinarea cazurilor 2 şi 3

Dacă se realizează simultan cazurile 2 şi 3, corecția se face prin adunarea lui 06 pentru cazul 3 şi adunarea lui 60 pentru cazul 2; aşadar, se adună în total 66:

	0101 1000+	(58 în BCD)
	0101 1001	(59 în BCD)
	1011 0001	(rezultat greşit, czurile 2 şi 4)
Corecție:	0110 0110	(se adună 66 în BCD)
	1 0001 0111	(117 în BCD, rezultat corect)

Concluzie. Când se execută adunări în cod BCD, se apelează imediat instrucțiuni de corecție a erorilor; acestea acționează în funcție de:

- indicatorul de transport (Carry Flag);
- indicatorul de semitransport (Auxiliary Carry) ;
- prezența codurilor inexistente în BCD;

În funcție de caz, corecția se face prin adăugarea unuia din numerele: 00, 06, 60, 66. În limbaj de asamblare (Intel 8086, 80x86, . . .) instrucțiunile apelate sunt:

- **DAA** (*Decimal Adjust on Addition*), ajustare zecimală la adunare;
- **DAS** (*Decimal Adjust on Substraction*), ajustare zecimală la scădere, caz în care erorile care apar sunt identice.

Corecția constă în adăugare de valori la adunare sau prin scăderea aceloraşi valori la scădere, acestea din urmă fiind tot adunări dar în complement față de 2.

Detecția codurilor inexistente (în BCD), reprezentând depăşiri de capacitate se face astfel: pentru digitul superior, dacă $\mathbf{b_7=1}$ şi simultan $b_5=1$ sau $b_6=1$; pentru digitul inferior, dacă $\mathbf{b_3=1}$ şi simulta $b_1=1$ sau $b_2=1$;

(am considerat notarea biților astfel: $\mathbf{b_7}\ b_6\ b_5\ b_4\ \ \mathbf{b_3}\ b_2\ b_1\ b_0$)

Tab. 8 : Codul ASCII (***A**merican **S**tandard **C**ode for **I**nformation **I**nterchange*)

0		32	?	64	@	96	`	128	Ç	160	à	192	└	224	α
1	□	33	!	65	A	97	a	129	ü	161	í	193	┴	225	ß
2	L	34	"	66	B	98	b	130	é	162	ó	194	┬	226	Γ
3	♥	35	#	67	C	99	c	131	â	163	ú	195	├	227	π
4	♦	36	$	68	D	100	d	132	ä	164	ñ	196	─	228	Σ
5	♣	37	%	69	E	101	e	133	à	165	Ñ	197	┼	229	σ
6	♠	38	&	70	F	102	f	134	å	166	ª	198	╞	230	µ
7	b	39	'	71	G	103	g	135	ç	167	º	199	╟	231	t
8	T	40	(	72	H	104	h	136	ê	168	¿	200	╚	232	Φ
9		41	)	73	I	105	i	137	ë	169	⌐	201	╔	233	Θ
10		42	*	74	J	106	j	138	è	170	¬	202	╩	234	Ω
11	J	43	+	75	K	107	k	139	ï	171	½	203	╦	235	δ
12	K	44	,	76	L	108	l	140	î	172	¼	204	╠	236	∞
13		45	-	77	M	109	m	141	ì	173	¡	205	═	237	φ
14	N	46	.	78	N	110	n	142	Ä	174	«	206	╬	238	ε
15	¤	47	/	79	O	111	o	143	Å	175	»	207	╧	239	∩
16	▶	48	0	80	P	112	p	144	É	176	▒	208	╨	240	≡
17	◀	49	1	81	Q	113	q	145	æ	177	▓	209	╤	241	±
18	↕	50	2	82	R	114	r	146	Æ	178	■	210	╥	242	≥
19	!!	51	3	83	S	115	s	147	ô	179	│	211	╙	243	≤
20	¶	52	4	84	T	116	t	148	ö	180	┤	212	╘	244	⌠
21	§	53	5	85	U	117	u	149	ò	181	╡	213	╒	245	⌡
22	_	54	6	86	V	118	v	150	û	182	╢	214	╓	246	÷
23	Q	55	7	87	W	119	w	151	ù	183	╖	215	╫	247	≈
24	↑	56	8	88	X	120	x	152	ÿ	184	╕	216	╪	248	°
25	↓	57	9	89	Y	121	y	153	Ö	185	╣	217	┘	249	∙
26	→	58	:	90	Z	122	z	154	Ü	186	║	218	┌	250	·
27	←	59	;	91	[	123	{	155	¢	187	╗	219	█	251	√
28	S	60	<	92	\	124	|	156	£	188	╝	220	▄	252	η
29	↔	61	=	93	]	125	}	157	¥	189	╜	221	▌	253	²
30	▲	62	>	94	^	126	~	158	₧	190	╛	222	▐	254	■
31	▼	63	?	95	_	127		159	ƒ	191	┐	223	▀	255	

Codul ASCII (***A**merican **S**tandard **C**ode for **I**nformation **I**nterchange*), este un cod binar alfanumeric, pe 8 biți, care realizează reprezentarea în binar a caracterelor imprimabile sau nu. Codul ASCII a fost definit de Institutul American de Standarde. La origine pe 7 biți (128 de simboluri), a fost dezvoltat de IBM la 8 biți, rezultând 256 simboluri (codul fiecărui simbol este reprezentarea în binar a numărului său de ordine)