Глава 2

Представљање података

За представљање података у меморији рачунара користе се различити формати зависно од тога о каквим се подацима ради и које ће се операције у процесу обраде подататка над њима извршавати. Заједничко за све формате је да се подаци у меморији рачунара региструју у облику бинарних низова за чије се генерисање у основи користе два општа концепта представљања података: бинарни бројни системи и бинарни кодови.

2.1 Бројни системи

Под бројним системом подразумевамо скуп правила дефинисаних за представљање бројних вредности података. У пракси данас најширу примену имају позициони бројни системи у које се сврстава и декадни систем бројева који користимо у свакодневном раду са бројевима. Заједничко за позиционе бројне системе је да се за представљање бројних вредности користи одређени скуп цифара, при чему вредност коју означава одређена цифра зависи од позиције на којој се у запису бројева налази. На пример, у декадном бројном систему користимо цифре: 0,1,2,3,4,5,6,7,8 и 9 али у броју 777.7 цифра 7 означава четири различите вредности: у првом сличају је то седам стотина (7×100), у другом седам десетиц (7×10), у трећем седам јединица (7), а у четвртом седам десетих делова јединице (7×0.1).

Ако једна цифра има увек исту вредност, без обзира на ком

се месту у запису налази, онда је то непозициони бројни систем. Типичан пример је римски бројни систем са цифрама: I,V,X,C,D,M.

У општем случају запису броја у позиционом бројном систему са основом N, где је N број различитих цифара бројног система, облика

$$(X)_N = x_n x_{n-1} \cdots x_0, x_{n-1} \cdots x_m, \tag{2.1}$$

одговара бројна вредност одређена релацијом

$$\begin{aligned}(X)_N &= \sum_{i=-m}^{n} x_i N^i \\ &= x_n N^n + x_{n-1} N^{n-1} + \cdots + x_0 + x_{-1} N^{-1} + \cdots + x{-m} N^{-m},\end{aligned} \tag{2.2}$$

где су са x_i означене цифре бројног система

У рачунарској техници значајну примену имају различити позициони бројни системи међу којима су најзначајнији: бинарни, октални и хексадецимални. У свим овим системима користе се иста правила за представљање бројева али се разликују скупови цифара помоћу којих се изражавају бројне вредности. У бинарном бројном систему користе се само цифре 0 и 1, у окталном скуп цифара $\{0,1,2,3,4,5,6,7,8\}$, а у хексадецималном скуп: $\{0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F\}$. Утабели 2.1 дати су неки примери декадних бројева и одговарајући еквиваленти у бинарном, окталном и хексадецималном бројном систему.

Табела 2.1: Примери бројева у различитим бројним системима

Декадни	Бинарн	Октални	Хексадецимални
1	1	1	1
7	111	7	7
8	1000	10	8
9	1001	11	9
15	1111	17	F
16	10000	20	10
17	10001	21	11

Представљање бројева у хексадецималном бројном систему се врло широко користи у рачунарској техници, нарочито када треба штампати или приказати бинарне садржаје, јер су записи компактнији и лакше се прате.

Пошто се у обичном животу најчешће користи децимални бројни систем, потребно је при уношењу нумеричких података у рачунар претворити децималне бројеве у бинарне. Такође је потребно по завршетку обраде података у рачунару излазне резултате из бинарног претворити у децимални облик.

У случају конверзије децималних бројева у бинарни облик посматраћемо одвојено конверзију целих и разломљених бројева. Конверзија целог децималног броја N у бинарни број може се применити применом следећег алгоритма:

1. Испитати да ли је N паран број,
2. а) Ако је N непаран, записати 1 у резултат и формирати нову вредност $N = N - 1$. Скок на 3.
   б) Ако је N паран, записат 0 у резултат.
3. Наћи нову вредност дељењем N из корака 2 са 2.
4. а) Ако је $N > 1$, вратити се на корак 1 и поновити поступак,
   б) ако је $N = 1$ уписати 1 у резултат.

При уписивању јединица и нула у резултат прво се уписује *бит најмање тежине*(енгл. Least Significant Bit - LSB) а остали се уписују лево од њега. Последњи уписани бит је *бит највеће тежине* (енгл. Most Significant bit -MSB). Добијени број је бинарни еквивалент децималног броја N.

Пример 2.1 *Конверзија децималног броја* 109_{10} *у бинарни број. Поступак конверзије се може приказати табеларно на следећи начин*

Број	Активност	Бит	
109	Непаран - одузми 1	1	LSB
108	Подели са 2		
54	Паран - подели са 2	0	
27	непаран - одузми 1	1	
26	Подели са 2		
13	Непаран - одузми 1	1	↓
12	Подели са 2		
6	Паран-подели са 2	0	
3	Непаран - одузми 1	1	
2	Подели са 2		
1	Крај алгоритма	1	MSB

Дакле, $109_{10} = 1101101_2$.

При конверзији правог децималног разломка користи се сличан алгоритам само процес почиње одређивањем бита највеће тежине, односно оног који се налази непосредно иза тачке која раздваја целобројни од разломљеног дела броја. Алгоритам за конверзију се сацоји само од два корака:

1. Формирати нови број N множењем N са 2.
2. а) Ако је $N > 1$, уписати 1 у резултат и одузети 1 од N. Скок на корак 1.

   б) Ак је $N < 1$, уписати 0 у резултат. Скок на корак 1

Поступак се завршава када се добије $N = 0$.

Пример 2.2 *Конверзија правог децималног разломка* 0.78125_{10} *у бинарни разломак. Поступак конверзије се изводи на следећи начин:*

Број	Активност	Бит	
0.78125	Помножи са 2		
1.5625	Одузми 1	1	MSB
0.5625	Помнож са 2		
1.125	Одузми 1	1	
0.125	Помножи са 2		↑
0.25	Помножи са 2	0	
0.5	Помножи са 2	0	
1.0	Одузми 1	1	LSB
0	Карј алгоритма		

Дакле, $0.78125_{10} = 0.11001_2$.

Треба приметити да се осим оваквог завршетка процеса конверзије може десити да се процес конверзије не заврши, јер се група битова понавља. Како сваки нови бит вреди половину предходног, конверзија се може прекинути када се процени да је постигнута задовољавајућа тачност.

Ако треба извршити конверзију децималног броја са целобројним и разломљеним делом, сваки део се посебно конвертује, а резултат је бинарни број са целобројним и разломљеним делом.

Конверзија из бинарног система у хексадецимални се изводи тако што се врши груписање битова бинарног броја у групе по

4 бита почевши са десне стране - LSB. Затим се свака група представи цифром од 0 до F која представља вредност групе. Конверзија из хексадецималног у бинарни систем врши се тако што се свака хексадецимална цифра замени групом од четири бита. Хексадецимални бројни систем се користи за компактно представљање података код рачунарских система, јер се сваки бајт представља са две хексадецималне цифре.

2.1.1 Бинарни кôдови

Као што је познато, рачунар "разуме" само језик цифара. За представљање податка користимо скуп симбола $A_k = \{a_1, a_2, \cdots, a_k\}$ и сваком податку придружујемо реч (низ симбола) над азбуком A_k. Процес придруживања речи азбуке A_k подацима назива се кодирање, а сам скуп речи је скуп података. Обрнути процес у коме се основни скуп речи датог кода одеђује скуп података назива се распознавање кода или декодирање.

Кодирање налази широку примену у различитим областима људске делатности и представља један од основних концепата представљања подата у информатици. Ако се за представљање података користе речи исте дужине (са истим бројем симбола) ради се о равномерним кодовима, док у супротном говоримо о неравномерним кодовима. Декодирање у случају равномерних кодова је једноставније па се зати чешће користе.

Кодови код којих се за представљање података користе кодне речи дефинисане над азбуком која садржи само два симбола, на пр. $A_2 = \{0, 1\}$, називају се бинарним кодовима. Бинарни кодови налазе широку примену у информатици и рачунарској техници јер се једноставно пресликавају у одговарајуће бинарне меморијске садржаје. Посебно значајани су бинарно кодирани децимални бројеви (BCD) којима се представља скуп декадних цифара, али и многи стандардни кодови (ISO 7, EBCDIC) којима се представљају знатно шири скупови ненумеричких података.

Бинарно-децимални кодови у којима се скуп децималних цифара $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ представља одређеним бинарним низовима, формирају се према различитим принципима. У табели 2.2 приказани су неки од бинарно децималних кодова. BCD кôд 842 је познат и под називом природни бинарно-децимални кôд, NBCD

кôд, и има најширу примену у односу на све остале BCD кодове. Овај кôд се сврстава у класу тежинских кодова, код којих се кодирање врши тако што је свакој од бинарних цифара у кодној речи придружена нека тежина. У случају овог кода то су тежине 8, 4, 2 и 1, тако да се вредност (d) декадне цифре која одговара кодној речи $b_3b_2b_1b_0$ добија као збир производа бинарних цифара и одговарајућих тежина. Односно важи следећа релација

$$d = 8b_3 + 4b_2 + 2b_1 + 1b_0 \tag{2.3}$$

Поред тежинских постоје и BCD кодови који се формирају применом неких других правила. Кôд вишка 3, је сличан природном бинарном кôду и од њега се разликује само по томе што се свакој децималној цифри прво дода 3 па се онда резултар кодује NBCD кôдом. Тако је нула кодирана као цифра три у природном кôду, цифра један као цифра четири и тако редом.

Кôд два од пет састоји се од речи дужине пет у којима се увек појављују две јединице.

Табела 2.2: Бинарно декадни кодови

Цифре	8421	вишка 3	2 од 5	Грејов	Хафменов
0	0000	0011	00011	0000	0000000
1	0001	0100	00101	0001	1101001
2	0010	0101	00110	0011	0101010
3	0011	0110	01001	0010	1000011
4	0100	0111	01010	0110	1001100
5	0101	1000	01100	0111	0100101
6	0110	1001	10001	0101	1100110
7	0111	1010	10010	0100	0001111
8	1000	1011	10100	1100	1110000
9	1001	1100	11000	1000	0011001

Напоменимо да Грејов кôд није тежински кôд те није погодан за извођење аритметичких операција, али је користан код аналогно-дигиталних претварача, улазно-излазних и других уређаја. Главна карактеристика Грејовог кôда је да се сваки кодни низ за суседну цифру, предходну или следећу, разликује само у једном биту (једној позицији). Грејов кôд се може дефинисати за свих шеснаест бинарнух низова дужине 4 као и за бинарне низове са било којом константном дужином.

Од бинарно декадних кодова са више од 4 бита посебно је интересантан 7-битни Хафменов кôд, јер он спада у кодове са откривањем и исправљањем грешака. Наиме, ако при преносу кодног низа неке декадне цифре дође до грешке у једној бинарној позицији, пријемни уређај може сам открити грешку и одмах је "аутомацки" исправити. Ако се грешка јави у две или три бинарне позиције истовремено, грешка ће бити откривена али се не може исправити, о чему уређај сигнализира. Означимо позиције Хафменовог кода слева у десно у десно са $AB8C421$, где су A, B и C контролни битови, а 8, 4, 2 и 1 информациони битови, контролни битови се добијају из израза

$$\begin{aligned} c_A &= c_8 \oplus c_4 \oplus c_1 \\ c_B &= c_8 \oplus c_2 \oplus c_1 \\ c_C &= c_4 \oplus c_2 \oplus c_1 \end{aligned} \tag{2.4}$$

где је знаком $\oplus$ обележена бинарна *искључиво или операција*. О овој операцији биће говора касније у делу о логичким функцијама.

2.1.2 Бинарно кодовани децимални бројеви

Као што је познато, децимални бројни систем се користи у обичном животу док је бинарни бројни систем погодан за примену у рачунарима. Бинарно кодовани децимални број представља компромисно решење између ова два система. Наиме, свака децимална цифра вишецифреног децималног броја представља се одговарајућим бинарним бројем и тако се добија *бинарно кôдовани децимални број* (BCD).

Природни бинарни децимални кôд (NBCD) представља децималне цифре од 0 до 9 групом од четири бинарне цифре. Конверзија децималног броја у NBCD број је једноставна и очигледна, као и обрнута конверзија. Међутим, извођење рачунских операција са BCD бројевима је знатно компликованије него са правим бинарним бројевима, па се BCD бројни системи за представљање бинарних бројева мало користе. Задржали су се само код калкулатора, уређаја са нумеричким приказивањем и нумеричких тастатура. Када се користи BCD представа броја потребно је, ради извођења аритметичких операција, претворити такав број у прави бинарни број. После обраде бинарни бројеби се, ради приказа, могу вратити у BCD формат.

За конверзију бинарно кодованих децималних бројева (NBCD) бројева у праве бинарне бројеве постоји више алгоритама. Један од таквих алгоритама је *алгоритам са сабирањем*. Према том алгоритму се прво свакој бинарној цифри NBCD броја придружи одговарајући тежински фактор. Идући са десна у лево тежински фактори су: 1, 2, 4, 8, 10, 20, 40, 80, 100, 200, итд. Затим се изврши сабирање тежинских фактора који одговарају битовима у NBCD броју чија је вредност јединица. У Табели 2.3 дати су тежински фактори у бинарном формату.

Табела 2.3: Тежински фактори за конверзију NBCD бројева у бинарне бројеве

Позиција	Тежински фактор	Бинарна вредност
1	1	1
2	2	10
3	4	100
4	8	1000
5	10	1010
6	20	10100
7	40	101000
8	80	1010000
9	100	1100100
10	200	11001000
11	400	110010000
12	800	1100100000
13	1000	1111101000
14	2000	11111010000

Пример 2.3 *Као пример, посматрајмо процес конверзије броја* 231_{10} *у бинарни број помоћу описаног алгоритма са сабирањем. NBCD представа броја* 213_{10} *је*

$$213_1 = 0010\ 0001\ 0011_{BCD}$$

Јединице у NBCD броју су на позицијама 1, 2, 5 и 10. Према табели 2.3 тежине које одговарају овим позицијама су:

$$\begin{array}{r} 1 \\ 10 \\ 1010 \\ 1100100 \end{array}$$

Сабирањем наведених тежинских фактора добија се 11010101_2 *што је бинарни еквивалент броју* 213_{10}.

Један од алгоритама за конверзију бинарног у NBCD број је *алгоритам одузимања*. Користећи Табелу 2.3 нађе се највећи број из табеле који је мањи или једанк од бинарног броја. Тај број се одузме од бинарног броја. Поступак се понавља са остатком све док остатак не буде нула. На свакој позицији где је вршено одузимање упише се јединица док се остале позиције попуне нулама. Резултат је NBCD еквивалент бинарног броја.

Пример 2.4 *Као пример посматрајмо конверзију бинарног броја* 11010101_2 *у NBCD број. Поступак се може спровести на следећи начин:*

11010101	*Бинарни број*
11001000	-200 *(Позиција 10)*
00001101	*Остатак*
00001010	-10 *(Позиција 5)*
00000011	*Остатак*
00000010	-2 *(Позиција 2)*
00000001	*Остатак*
00000001	-1 *(Позиција 1)*
00000000	*Крај*

Дакле, конверзијом се добија да важи једнакост $11010101_2 = 0010\ 0001\ 0011_{NBCD}$. *Као што се види, конверзијом је добијен децимални број 213, што се и могло очекивати с обзиром на конверзију у обрнутом смеру који описан у предходном примеру.*

2.2 Представљање нумеричких података

Формат представљања података је основно својство на основу којег се могу разликовати различити типови података. Сви подаци се приликом регистровања у меморији рачунара своде на унапред дефинисане формате стандардне дужине. За стандардне дужине формата података од 8, 16, 32 и 64 бита користе се називи бајт, полуреч, реч и двострука реч, респективно.

2.2.1 Целобројни подаци

Цели бројеви се обично дефинишу као подаци типа INTEGER. За нихово представљанне на у речунару користи се нотација која је уобичајена у математици: низ цифара са или без знака. У меморији рачунара за представљање целих бројева се користи једна реч

дужине 32 бита. Крајње лева позиција је позиција знака броја, док се све остале користе за представљање апсолутне вредности броја. Позитивни бројеви се представљају тако што се директно преводе у бинарни бројни систем и у позицију знака се уписује цифра 0.

Потреба да се јединствени хардвер користи за операцију сабирање и одузимање довела је до представљања негативних бројева коришћењем комплемента броја. Комплемент броја може да буде потпун или непотпун. Непотпуни комплемент бинарног броја формира се тако што се свака цифра замени компелментарном цифром, односно нула се замени јединицом, а јединица нулом. Потпуни комплемент се добија када се непотпуном комплементу на позицији најмање тежине дода један. У случају и једног и другог комплемента броја на позицији знака броја налази се јединица.

У стандардном формату бројева негативни бројеви се представљају преко потпуног комплемета.

Пример 2.5 *Стандардни формат броја 183*

00000000 00000000 00000000 10110111

Стандардни формат броја −183

11111111 11111111 11111111 01001001

Нека је X m-троцифрени бинарни број

$$X = \pm x_{m-1}x_{m-2}\ldots x_1x_0. \tag{2.5}$$

Његов потпуни комплемент је

$$Y = y_m y_{m-1}\ldots y_1 y_0, \tag{2.6}$$

у коме је y_m цифра знака. Број X је одређен следећим изразом

$$X = -y_m 2^m + y_{m-1}2^{m-1} + y_{m-2}2^{m-2} + \cdots + y_1 2 + y_0 \tag{2.7}$$

Поред стандардног формата целих бројева дужине једне речи, користе се и неки скраћени формати од једне полуречи или једног бајта, али и проширени формати двоструке речи. Принцип регистровања бројева су исти као код стандардног целобројног формата.

Пример 2.6 *Дат је потпуни комплемент* $Y = 111010001$ *неког броја* X*. Наћи број* X*.*

$$X = -2^8 + 2^7 + 2^6 + 2^4 + 2^0 = -256 + (128 + 64 + 16 + 1) = -47$$

За представљање бројева може се користити и формат непокретне тачке који се састоји у томе да се од једне меморијске локације одређени број позицвија користи за представљање целобројног дела броја, а остатак за разломљени део. Позиција највеће тежине се одваја за знак броја. На слици 2.1 приказан је пример овог формата.

31	30 16	15 0
s	Цели део броја	Разломљени део броја

Сл. 2.1: Формат непокретне тачке

Формат непокретне тачке има ограничену примену због тога што знатно смањује опсег бројева који се могу представити у једној меморијској речи.

Правила за аритметичке операције која важе у декадном бројном систему могу се применити и у било ком другом природном бројном систему са основом q. Сабирање два броја врши се сукцесивним сабирањем цифара почев од цифре најмање тежине. При томе је

$$a_i + a_j = \begin{cases} a_k, \text{ где је } a_k = a_i + a_j & \text{ако је } a_i + a_k < q \\ 1a_k, \text{ где је } a_k = a_j + a_k - q & \text{ако је } a_i + a_j \geq q \end{cases} \tag{2.8}$$

где су a_i, a_j и a_k цифре бројног система са основом q. Сабирање два вишецифрена броја у позиционом бројном систему са основом q врши се према следећем алгоритму:

1. Сабрати две цифре најмање тежине. Ако постоји јединица преноса запамти се за следћу вишу позицију.
2. Сабрати две цифре цледеће више позиције и цифру преноса из предходне ниже позиције.

3. Ако има још позиција у којима није извршено сабирање ићи на корак 2.
4. Ако је извршено сабирање цифара у свим позицијама и не постоји пренос, поступак сабирања је завршен. Ако постоји пренос уписати 1 у следећу вишу позицију, чиме је поступак сабирања завршен.

Пример 2.7 *Сабирање бинарних бројева* 1100111 *и* 110110.

$$\begin{array}{r} 1100111 \\ +110110 \\ \hline 10011101 \end{array}$$

Одузимање бројева у позиционом бројном систему са основом q врши се по правилима која важе у декадном бројном систему. При томе се код рада са разломљеним бројевима тачке разломака пишу у истој позицији по вертикали. Одузимање два броја се врши сукцесивно одузимањем цифара почев од најниже позиције. При томе је

$$a_i - a_j = \begin{cases} a_i - a_j, & \text{ако је } a_i \geq a_j \\ q + a_i - a_j, & \text{ако је } a_i < a_j \end{cases} \tag{2.9}$$

где се за случај $a_i < a_j$ "позајмљује" једна јединица из више позиције која садржи q јединица ниже позиције.

Пример 2.8 *У бинарном бројном систему од бинарног броја* 10101.11 *одузети бинарни број* 1001.001

$$\begin{array}{r} 10101.\ \ 110 \\ -1001.\ \ 001 \\ \hline 1100.\ \ 101 \end{array}$$

Представљање негативних бројева у облику комплемента омогућује да се операција одузимања обави на сабирачима. Наиме, операција одузимања своди се на операцију сабирања комплемената бројева. При коришћењу потпуног коплемента при сабирању пренос из позиције знака се губи, док се при коришћењу непотпуног комплемента пренос из позиције знака накнадно додаје суми комплемената у позицији најмање тежине. При сабирању

комплемената сабирака може доћи до прекорачења капацитета коришћеног формата бројева. Критеријум за установљење прекорачења је следећи: ако је цифра преноса из позиције знака различита од цифре преноса у позицију знака дошло је до прекорачења. У том случају добијени резултат је погрешан. Исправан резултат се може добити применом дужег формата бројева.

Пример 2.9 *Применом потпуног комплемента израчунати (а) разлику бинарних бројева:* $c = 111_2 - 101_2$ *и (б) збир негативних бинарних бројева* $d = -100_2 - 110_2$*. За комплементе бројева усвојити дужину од четири бита. Вертикална црта* | *одваја пренос из позиције знака који*

prenos = 1|111	$c = 7$
c =1111	-5
+1011	$+2$
0010	

(а)

prenos = 1|000	$d = -4$
d =1100	-6
+1010	-10
0110	

(б)

се игнорише јер се за представљање бројава користе четири бита.

Разлика бројева је c *је позитивна са вредношћу* 10_2*, док је збир негативних бројева такође позитиван са вредношћу* 110_2*, што је погрешан резултат. До очигледне грешке је дошло због прекораченог капацитета коришћеног формата бројева. На то указују различите вредности преноса у позицију знака суме (0) и из позиције знака суме (1). Продужавањем формата бројева на пет цифара добија се исправан резултат:*

prenos = 1|1000	$d = -4$
d =11100	-6
+11010	-10
10110	

Комплемет је сада 10110*, што за суму даје исправну вредност* -1010_2*.*

У аритметици која користи потпуни комплемент, прекорачење које настаје као међурезултат сабирања више бинарних бројева,

неће проузроковати грешку у коначном резултату уколико је коначни резултат у задатом опсегу бројева.

Пример 2.10 *Коришћењем потпуног комплемента бинарних бројева израчунати алгебарски збир бројева* $x = 27 + 16 - 18$. *Комплементе бројева представљати са 6 бинарне цифре. Сабирањем бројева 27 и 16*

$x =$011011	$x =$27
010000	16
101011	43
101110	$-$18
1|011001	25

дошло је до прекорачења, јер је највећи број који се може представити са 6 бинарне цифре је 31 (једна цифра је резервисана за знак). Коначни резултатт је исправан.

Познато правило за множење које се користи у декадном бројном систему примењује се и за множење бројева у бинарном бројном систему. Разлика је једино у томе што се при множењу и сабирању користе правила која важе за бинарни бројни систем. Множење вишецифрених бинарних бројева врши се тако што се множеник множи са цифром из сваке позиције множиоца, почев од цифре најмање тежине, парцијални производи се потписују један испод другог, померени за једно место у лево, а затим саберу

Пример 2.11 *Множење бројева* 11001 *и* 110 *у бинарном бројном систему.*

$$\begin{array}{r} 11001 \times 110 \\ 00000000 \\ 00011001 \\ 00011001 \\ 10010110 \end{array}$$

Описани поступак множења бинарних бројева показује да се реализација бинарног множења у рачунару своди на вишеструко бинарно сабирање и померање за једну бинарну позицију. Дакле, за

множење бинарних бројева потребно је да аритметичко-логичка јединица рачунара садржи само сабирач и кола за померање.

Множење означених бинарних бројева, ако су негативни бројеви изражени у потпуном коплементу, може се обавити као множење неозначених бројева, водећи рачуна да бит знака има тежину -2^{n-1}. Из тог разлога се парцијалном производу, који потиче од бита највеће тежине множиоца, мора променити знак, односно последњи парцијални производ је потпуни комплемент множеника. Пошто у означеном бројном систему позиција бита за знак мора да буде дефинисана, то се унапред одређује број цифара производа, који мора бити већи или једнак збиру броја цифара множеника и множиоца. Свим парцијалним производима треба повећати број цифара на усвојени број цифара производа, водећи рачуна о знаку[1]. Приликом сабирања игнорише се евентуални пренос који је последица сабирања бита знака.

Наведена правила су показана у следћим примерима множења четвороцифрених бинарних бројева, а усвојено је да производ има осам цифара.

Пример 2.12 *Множење бројева* $0101_2 = 5_{10}$ *и* $1101_2 = -3_{10}$ *у бинарном бројном систему.*

$0101 \times 1101 =$ 00000101
0000000
000101
11011 потпуни копмпемент множеника
$11110001 = -15$

Ако је множилац позитиван, тада је последњи парцијални производ једнак нули па се може и изоатавити.

Пример 2.13 *Множење бројева* $1101_2 = -3_{10}$ *и* $0101_2 = 5_{10}$ *у бинарном бројном систему*

[1] Повећање броја цифара код позитивих бројева се сасоји у додавању нула, док се повећање броја цифара код негативних бројева састоји у додавању јединица. На пример, повећање броја цифара за позитиван број пет је $0101 = 00000101$, док за повећње броја цифара за негативан број пет је $1011 = 1111011$. У оба случаја бит највећ тежине је знак броја.

$$
\begin{array}{rl}
1101 \times 0101 = & 11111101 \\
& 0000000 \\
& \underline{111101\ \ } \\
& 1|11110001 = -15 \\
& \uparrow \text{пренос који се занемарује}
\end{array}
$$

Лако се може показати да је резултат позитиван ако су множеник и множилац негативни.

Пример 2.14 *Множење бројева* $1101_2 = -3_{10}$ *и* $1011_2 = -5_{10}$ *у бинарном бројном систему.*

$$
\begin{array}{rl}
1101 \times 1011 = & 11111101 \\
& 1111101 \\
& 000000 \\
& \underline{00011\ \ } \text{потпуни копмпемент множеника} \\
& 10|00001111 = 15 \\
& \uparrow \text{пренос који се занемарује}
\end{array}
$$

Најједноставније правило за дељење бинарних бројева је исто као и за дељење децималних бројева. Број који образује цифре највеће тежине дељеника се подели са делиоцем, остаку се придружује следећа цифра нељеника и процедура се наставља све док остатак не буде мањи од делиоца. И у децималном и у бинарном систему дељење може да се настави множећи остатак основом бројног система (придружујући остатку 0), а у количници цела места треба одвојит тачком од разломачких.

Број корака дељења ограничен је бројем позиција за количник, због чега се дељење најчешће врши заокруживањем, тј. приближно. Укажимо још и на то, да описани поступак дељења захтева само операцију бинарног одузимања и померање бинарних бројева за једно место у лево или у десно.

Наведено правило које важи за неозначене бинарне бројеве, илустровано је у следећм примеру.

Пример 2.15 *Дељење броја* $281_{10} = 100011001_2$ *бројем* $13_{10} = 1101_2$ *у бинарном бројном систему.*

Дељење означених бинарних бројева је нешто сложеније, међутим постоје правила и алгоритми за дељење бројева чије су негативне вредности изражене у потпуном комплементу.

2.2.2 Формат покретне тачке

За рад са реалним бројевима у обради података се најчешће користи тип података REAL (помекад се означава и са FLOAT) који се заснива на представљању података у формату покретне тачке. Формат покретне тачке може бити са децималном тачком или у експоненцијалном облику.

Експоненцијални облик формата покретне тачке[2] представља се тако што се најпре преведе у бинарни бројни систем и запис бројне вредности трансформише у експоненцијални облик тако да је

$$X = M \cdot 2^E, \tag{2.10}$$

где је m мантиса, $1 \leq M = m.f < 2$, а E је експонент броја. Број се представља тако што се у одређену меморијску локацију уписује цифра знака броја, цифре мантисе и цифре експонента броја. Експонент се представља у поларизованом облику $e = E + P$, где је P поларизација или помак величине $P = 2^{k-1} - 1$, а k је број бинарних цифара експонента. Овом трансформацијом се вредности експонента E из опсега $[-2^{k-1} - 1, 2^{k-1}]$ при поларизацији $P = 2^{k-1} - 1$ пресликавају у опсег вредности $[0, 2^k - 1]$ поларизованог експонента e. Пошто цели део нормализоване мантисе мора бити једнак јединици, могуће је изоставити ову јединицу, и од мантисе чувати само разломак. Зато се $m = 1$ у нормализованој мантиси $M = m.f$ назива имплицитна или скривена јединица. Ово решење које је могуће само само за основе експонета 2, омогућује да се на рачун скраћивања поља мантисе мантисе продужи поље експонета за један бит, и тиме повећа опсег представљања бројева.

За представљање реалног броја у меморији рачунара користи се стандардни формат покретне тачке дужине једне меморијске речи,

[2]Стандарди за аритметику са покретном тачком су: IEEE 754-85, IEC 559 и JUS N.R5.211

```
100011001 : 1101 = 10101   Количник
−1101
  1001
 −0000
  10010
 −1101
   1010
  −0000
   10101
  −1101
    1000   Целобројни остатак
```

као што је приказано на слици 2.2, са пољима: знак, поларизовани експонент и мантиса. Дужина ових поља је 1, 8 и 23 бита.

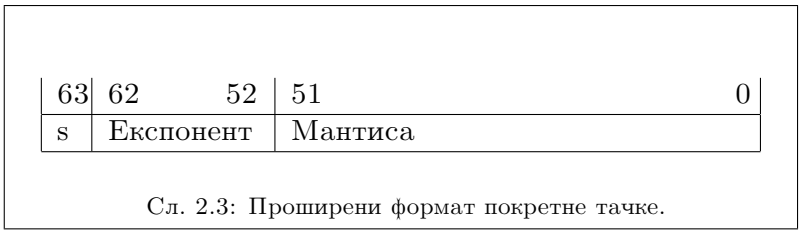
31	30 23	22 0
s	Експонент	Мантиса

Сл. 2.2: Стандардни формат покретне тачке.

Поред стандардног формата покретне тачке користи се и проширени формат који се обично означава као DOUBLE PRECISION. У програмима за обраду података ови подаци се увек представљају у експоненцијалном облику са словом D у запису: 1.009 D 01, 1.55 D −0.2.

За представљање једног броја у овом формату користе се две меморијске речи тако да се формирају бинарни низови дужине 64 бита, као што је приказано на слици 2.3. Запис броја се формира према истим принципима као код стандардног формата REAL с том разликом да су поља експонента и мантисе шира. То омогућава да се представе бројеви из ширег опсега вредности и са већом тачношћу.

Вредности експонента $e = 255$ код стандардног формата покретне тачке и $e = 2047$ код проширеног формата покретне тачке, је резервисана за кодирање бесконачне вредности ($\pm\infty$) и небројева *NaN* (Not a Number).



Сл. 2.3: Проширени формат покретне тачке.

Правила за интерпретацију стандардног формата покретне тачка су следећа:

- ако је $e = 255$ и $f \neq 0$ тада је $v = NaN$;
- ако је $e = 255$ и $f = 0$ тада је $v = (-1)^s \cdot \infty$;
- ако је $a < e < 255$ и тада је $v = (-1)^s \cdot 2^{e-127} \cdot 1.f$;
- ако је $e = 0$ и $f \neq 0$ тада је $v = (-1)^s \cdot 2^{-126} \cdot 0.f$ (денормализовани број);
- ако је $e = 0$ и $f = 0$ тада је $v = (-1)^s \cdot 0 = 0$.

Слична правила се могу формирати за интерпрентацију проширеног формата покретне тачке водећи рачуна да поларизација за овај формат износи $P = 1023$.

Пример 2.16 *Дат је 32-битни податак*

$$1|10000001|0100000\ 00000000\ 00000000$$

Који број представља према стандарду IEEE 754?

Из датог податка је $e = 1000001_2 = 129_{10}$*, па дати податак представља број* $v = 1.fe^{e-127}$*. За поларизовани експонент* $e = 129$ *неполаризовани експонет је* $E = e - 127 = 129 - 127 = 2$*. Такође, из датог податка је* $f = 0.01_2$ *а* $M = 1.f = 1.01_2$*. Дакле,* $v = M2^E = 1.01 \cdot 2^2 = 101_2 = 5_{10} = 5$.

Сабирање и одузимање бројева са покретном тачком обавља се свођењем оба сабирка на исти експонент. При томе већи сабирак мора остати нормализован па се експонент мањег сабирка изједначава са експонентом већег сабирка. Наравно, то мора бити праћено померањем мантисе мањег сабирка у десно за број бинарних позиција одређен рзликом експонената већег и мањег сабирка.

Овим померањем мањи сабирак постаје денормализован. Дакле, за збир бројева $A = M_A \cdot 2^{M_A}$ и $B = M_B \cdot 2^{M_B}$ може се писати

$$\begin{aligned} C = M_C \cdot 2^{e_C} &= M_A \cdot 2^{e_A} \pm M_B \cdot 2^{e_B} \\ &= (M_A \cdot 2^{e_A - e} \pm M_B \cdot 2^{e_B - e}) \cdot 2^e \end{aligned}$$

где је e експонет већег сабирка и у исто време експонент збира, $e_C = e$. По потреби се може нормализовати вредност збира C.

Приликом множења бројева A и B треба најпре сабрати експоненте, а затим одузети поларизацију: $e_C = a_A + e_B - P$. У следећм кораку се множе мантисе и одређује знак производа. По потреби нормализвати вредност производа, свести вредност мантисе на 23 цифре иза бинарне тачке. Одузимање поларизације је неопходно да би се експонент заджао у поларизованом облику.

Нека је после нормализације у неком кораку аритметичке операције добијена вредност мантисе $M = 1.f_{-1}f_{-2}\ldots f_{-23}f_{-24}$ коју треба свести на 23 бинарне цифре. Свођење разломка са 24 на 23 цифре може се извести на два начина: одсецањем и заокруживањем.

Одсецање је поступак којим се одбацују додатне цифре разломка тако да се добија $M = 1.F f_{-1}f_{-2}\ldots f_{-23}$. Максимална вредност грешке која се том приликом чини је приближно једнака вредности јединице у позицији цифре $f_{-23}(2^{-23})$. За одсецање кажемо да је полазизовано јер је грешка увек истог (позитивног) знака.

Заокруживање је поступак којим се мантиса облика $M = 1.f_{-1}f_{-2}\ldots f_{-23}f_{-24}$ замењује вредношћу добијеном додавањем цифре f_{-24} у позицију цифре f_{-23}. При $f_{-24} = 0$ заокруживање се своди на одсецање. У овом случају заокруживањем се прави максимална греска која је приближно једнака половини јединице у позицији цифре $f_{-23}/2(2^{-24})$. При $f_{-24} = 1$ заокруживањем се прави грешка која приближно износи $-f_{23}/2(-2^{-24})$. Дакле, грешка која се прави заокруживањем није поларизована већ је означена.

Аритметичке операције бројева са бројевима који су представљени покретном тачком обављају се по посебним правилима. Њихова реализација захтева специфичне поступке и аритметичке блокове и сложенија је за целобројну аритметику или аритметику са непокретном тачком. Осим тога и време извршења аритметичких операција са покретном тачком је дуже.