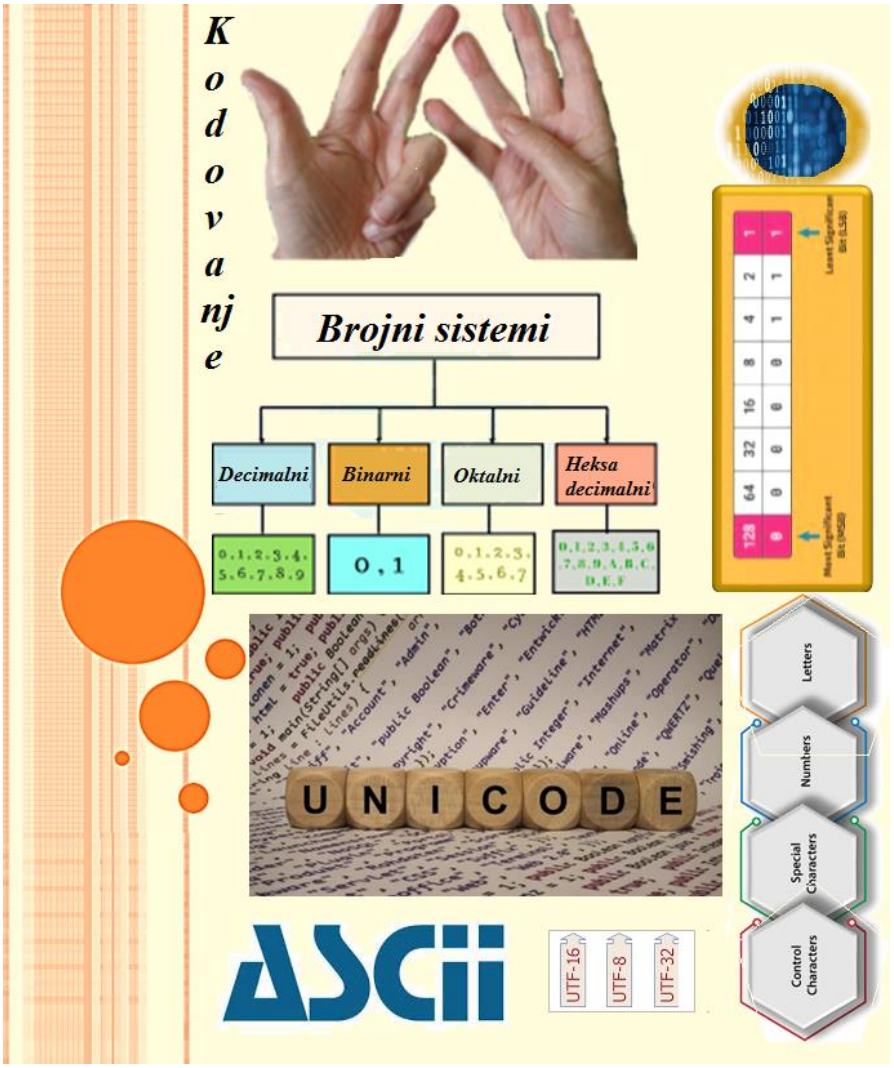
Miroslav Mihaljišin

Predstavljanje podataka u računaru

Brojni sistemi i kodovanje



Drugo online izdanje
BL, mojTim,2023

Brojni sistemi i kodovanje

Sadržaj 2
1 **Brojni sistemi 3**
1.1 Vrste brojnih sistema 3
1.2 Dekadni sistem 4
1.3 Binarni sistem 4
1.4 Oktalni sistem 5
1.5 Heksadecimalni sistem 6
1.6 Primjeri konverzija iz jednog brojnog sistema u drugi 7
1.6.1 Konverzija decimalnog broja u binarni 7
1.6.2 Konverzija oktalnih brojeva u decimalne i obrnuto 8
1.6.3 Konverzija binarnih brojeva u oktalne i obrnuto 9
1.6.4 Konverzija binarnih brojeva u heksadecimalne i obrnuto 9
1.6.5 Konverzija decimalnih u heksadecimalne i obrnuto 10
2 Osnovne aritmetičke operacije u binarnom brojnom sistemu 11
2.1 Primeri aritmetičkih operacija 12
3 Predstavljanje i označavanje binarnih brojeva 14
3.1 Neoznačeni cijeli binarni brojevi 14
3.2 Označeni cijeli binarni brojevi sa bitom znaka 15
3.3 Označavanje brojeva korišćenjem komplementa 16
3.4 Sabiranje i oduzimanje brojeva zapisanih u komplementu dvojke 18
3.5 Predstavljanje realnih brojeva sa fiksnom decimalnom tačkom 20
3.6 Predstavljanje brojeva sa pokretnom decimalnom tačkom 20
4 Korišćenje kodova za zapis podataka 25
4.1 Kodovi za zapis brojeva 25
4.2 BCD kod 25
4.2.1 Sabiranje brojeva zapisanih u BCD kodu 26
4.3 Kodovi za zapis teksta kod računara 27
4.3.1 ASCII-ošišana latinica 28
4.3.2 Kodne strane 30
4.3.3 Unikod (unicode) 32
4.3.4 UTF-8 33
Izvori i reference 35

1 Brojni sistemi

Brojni sistem je sistem pomoću kojeg se predstavljaju brojevi. Brojni sistem se uvijek sastoji iz baze b i skupa simbola koje nazivamo ciframa.

1.1 Vrste brojnih sistema

Razlikujemo dvije osnovne vrste brojnih sistema:

- nepozicione (netežinske) i
- pozicione (težinske)

Najpoznatiji netežinski brojni sistem je rimski. Ali to su i neki drugi sistemi drevnih i izumrlih naroda npr. egipatski brojni sistem, vavilonski brojni sistem i brojni sistem Maja.

Pozicioni brojni sistemi su sistemi u kojima se težina cifre, odnosno njen udio u vrijednosti broja određuje na osnovu njene pozicije u broju; što veća pozicija to je veći i udio u vrijednosti broja.

Pozicioni brojni sistem se može prikazati formulom:
$(...a_3,a_2,a_1,a_0.a_{-1},a_{-2},...)_b=...+a_3\cdot b^3+a_2\cdot b^2+a_1\cdot b^1+a_0\cdot b^0+a_{-1}\cdot b^{-1}+a_{-2}\cdot b^{-2}+...$
gdje je b baza (ili osnova) brojnog sistema, a a_i cifre brojnog sistema.

U opštem slučaju b može biti bilo koji broj različit od nule, ali kod standardnih brojnih sistema za b se uzima prirodni broj veći od 1. U slučaju standardnih brojnih sistema cifre se biraju tako da uzimaju vrijednosti prirodnih brojeva između 0 i b (baze). **Baza** (base ili radix) **brojnog sistema je broj simbola u sistemu.**
Tako dobijamo standardne brojne sistema binarni (b=2), kvarterni (b=4), oktalni (b=8) i heksadecimalni (b=16).

Najpoznatiji pozicioni brojni sistem je dekadni i on ima bazu 10.
U računarima se najčešće koristi binarni brojni sitem sa bazom 2.

Za razumijevanje brojnih sistema potrebno je napraviti razliku između broja, kao prirodno-matematičkog pojma, i njegovog predstavljanja. Npr. broj dva (dvije kruške) se može predstaviti kao 2, u slučaju da koristimo dekadni sistem, ali i kao 10 ako se koristi binarni sistem.

1.2 Dekadni sistem

Dekadni brojni sistem je pozicioni brojni sistem sa bazom 10 u kojem se zapis sastoji od cifara 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.
Svaki broj se može predstaviti kao zbir eksponenata broja 10.
Bilo koji **n**-cifreni cijeli dekadni broj može predstaviti nizom:

$$A = a_n 10^n + a_{n-1}10^{n-1} + \cdots + a_1 10^1 + a_0 10^0$$

gdje koeficijenti a_0, a_1, a_2,....,a_n predstavljaju cifre dekadnog broja i imaju vrijednost od 0 do 9.

Težinska vrijednost je vrijednost cifre u zavisnosti od njenog mjesta u nekom broju. To znači da se prvobitna vrijednost cifre množi brojem 10 za svaku poziciju lijevo od posljednje cifre u zapisu broja.

Slično cjelobrojnim, predstavljaju se i decima brojevi. Vrijednosti iza decimalnog zareza množe se sa odgovarajućim eksponentom kome se pridružuje negativna vrijednost.
Kao ilustraciju možemo dati primjer dekadnog broja 4275,59 koji se može predstaviti kao: $4\cdot10^3+2\cdot10^2+7\cdot10^1+5\cdot10^0+5\cdot10^{-1}+9\cdot10^{-2}$

1.3 Binarni sistem

Binarni ili dualni sistem je prvi koristio Leibniz u 17. vijeku. Danas je pored dekadnog sistema ovo najrasprostranjeniji brojni sistem.
Binarni sistem je sistem na bazi 2, u kome se za označavanje brojeva koriste isključivo dvije cifre (0 i 1).
Bilo koji **cijeli binarni broj** se može predstaviti kao niz:

$$A = a_n 2^n + a_{n-1}2^{n-1} + \cdots + a_1 2^1 + a_0 2^0$$

gdje koeficijenti a_0, a_1, a_2,....,a_n imaju vrijednost 0 ili 1 i predstavljaju cifre binarnog broja. Svaki član u ovom nizu ima dvostruku težinu u odnosu na prethodni član, što je pokazano Tabeli 1.

težinske vrijednosti	...	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-3}	...
dekadni ekvivalent	...	1024	512	256	128	64	32	16	8	4	2	1	0,5	0,25	0,125	0,0625	...

Tabela 1 Težine cifri u u binarnom brojnom sistemu

Zadatak: **Rastaviti broj** 10011101 prema definiciji predstavljanja binarnog broja kao niza i odrediti koji je to dekadni broj.

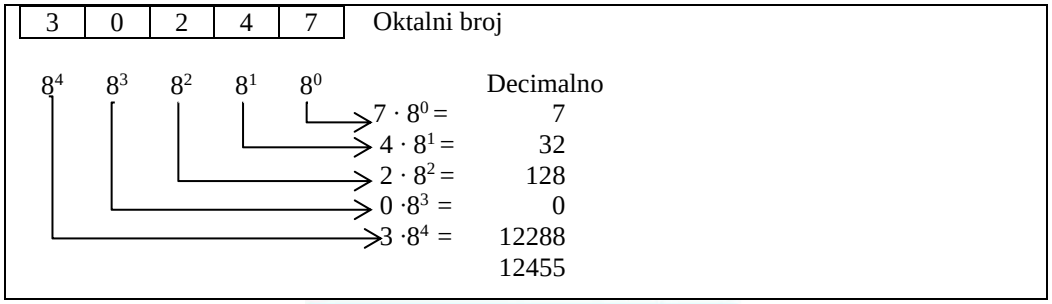
Rješenje:

$1 \cdot 2^7 + 0 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 128 + 16 + 8 + 4 + 1 = 157$

Ekvivalentno predstavljanju cijelih brojeva, vrijednost binarnih brojeva iza decimalnog zareza predstavlja se korištenjem negativnih eksponenata.

1.4 Oktalni sistem

Oktalni brojni sistem je pozicioni brojni sistem sa bazom 8. To znači da se svaki broj može predstaviti kao zbir eksponenata broja 8. Oktalni sistem se često primjenjuje u računarstvu i informatici.



Binary	Octal
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

Ilustracija 1 Primjeri pretvaranje oktalnog broja u decimalni i binarni

Oktalni broj može se dobiti iz binarnog tako da se binarne cifre grupišu po tri i potom zamjene oktalnim.

1.5 Heksadecimalni sistem

Heksadecimalni brojni sistem je težinski brojni sistem sa bazom 16. Kao cifre se koriste: 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F.
Cifre 0-9 imaju istu vrijednost kao u dekadnom sistemu. Broj 10 se piše kao A, broj 11 kao B, 12-C, 13-D, 14-E i 15-F.
n-cifreni cijeli brojevi u hexadecimalnom brojnom sistemu sa pozicionom notacijom se se mogu predstaviti kao niz:

$$A = a_n 16^n + a_{n-1} 16^{n-1} + \cdots + a_1 16^1 + a_0 16^0$$

gdje koeficijenti a_0, a_1, a_2,....,a_n imaju vrijednost 0 do 15 i predstavljaju cifre heksadecimalnog brojnog sistema.

Heksadecimalni brojni sistem ima veliku praktičnu vrijednost jer mu je baza eksponent broja 2, koji je osnova binarne logike. Svaka cifra u heksadecimalnom sistemu mijenja 4 uzastopne cifre binarnog sistema.
Heksadecimalni sistem se koristi u računarstvu u kombinaciji sa binarnim sistemom jer omogućava jednostavnu konverziju. 4 mjesta u binarnom sistemu zauzimaju samo jedno mjesto u heksadecimalnom sistemu.
Na primjer **binarni** broj 1110 0001 1111 1101 se **heksadecimalno kraće** prikazuje kao E1FD.

DECIMALNI	BINARNI	OKTALNI	HEKSADECIMALNI
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Tabela 1 Najčešći brojni sistemi

1.6 Primjeri konverzija iz jednog brojnog sistema u drugi

Često se javlja potreba za pretvaranjem (konverzijom) brojeva iz jednog brojnog sistema u neki drugi brojni sistem.
Ukoliko je baza brojnog sistema u kome se nalazi broj kog želimo konvertovati, veća od baze broja u koji sistem želimo izvršiti konverziju, najjednostavnije ćemo to učiniti tako što broj dijelimo bazom brojnog sistema u kojeg ga pretvaramo sve dok količnik ne bude 0 a potom prepišemo ostatke dijeljenja unazad. Ovako opisana procedura naziva se **dijeljem sa modulom**, gdje je modul (MOD) ostatak cjelobrojnog dijeljenja.
Za **obrnutu konverziju brojeva** (*iz sistema koji koristi manju bazu u sistem koji ima veću bazu*) koristiti se **direktno sumiranje (rastavljanje) članova**, kako je ranije objašnjeno na primjeru rastavljanja binarnog broja.

1.6.1 Konverzija decimalnog broja u binarni

Dekadni broj se dijeli sa 2 (2 je baza binarnog brojnog sistema), a ostatak dijeljenja posebno zapisuje. Konvertovani broj se dobija kada se napišu cifre ostatka obrnutim redom od onog kako su dobijene.
Zadatak: $157_{10}=?_2$

157	:	2	=	78	↑	(1)
78	:	2	=	39	|	(0)
39	:	2	=	19	|	(1)
19	:	2	=	9	|	(1)
9	:	2	=	4	|	(1)
4	:	2	=	2	|	(0)
2	:	2	=	1	|	(0)
1	:	2	=	0		(1)

Rješenje: $157_{10}=10011101_2$

Decimalni brojevi manji od jedinice (brojevi sa decimalnim zarezom manji od 1) konvertuje se u binarni tako što se razlomljeni dio množi sa 2 i postupak nastavlja dok razlomljeni dio ne postane nula.
Primjer konverzije broja 0,375

0,375	*2	=0,75	=0,75	+0
0,75	*2	=1,5	=0,5	+1
0,5	*2	=1	=0	+1

Rješenje $(0,375)_{10}=(0,011)_2$

Konverzija brojeva koji nisu cijeli ne može uvijek izvesti do kraja, što podrazumijeva primjenu aproksimacija.
Primjer:

0,792	*2	=1,584	=0,584	+1
0,584	*2	=1,168	=0,168	+1
0,168	*2	=0,336	=0,336	+0
0,336	*2	=0,672	=0,672	+0
0,672	*2	=1,344	=0,344	+1
0,344	*2	=0,688	=0,688	+0
0,688	*2	=1,376	=0,376	+1

Rješenje $(0{,}792)_{10}=(0{,}1100101...)_2$

Aproksimacija ne znači da je binarni sistem manje precizan. To znači binarni sistem zahtijeva veći broj cifara za izražavanje određene veličine željenom preciznošću.
Konverzija mješovitih decimalnih brojeva vrši se tako što se **posebno konvertuje cjelobrojni dio, a posebno dio sa razlomljenim vrijednostima**, pa se dobijeni rezultati sabiraju.
Za **obrnutu konverziju binarnih brojeva u decimalne** može se koristiti **direktno sumiranje članova**, kako je ranije objašnjeno na primjeru rastavljanja binarnog broja.

Ovo ćemo još jednom pokazati na primjeru gdje se traži konverzija binarnog broja $(110011101{,}01)_2$
Rješenje: Svakoj poziciji se pridruži vrijednost eksponenta koja se množi sa vrijednošću binarne cifre (0 ili 1):
$1\cdot2^8+1\cdot2^7+0\cdot2^6+0\cdot2^5+1\cdot2^4+1\cdot2^3+1\cdot2^2+0\cdot2^1+1\cdot2^0$ **,** $0\cdot2^{-1}+1\cdot2^{-2}$
256+128+16+8+4+1**,** +1/4=413,25

1.6.2 Konverzija oktalnih brojeva u decimalne i obrnuto

Konverzija koristi isti metod objašnjen kod binarnih brojeva, *samo je različita baza i cifre.*
Primjer: $157_{10}=?_8$

157	:	8	=	19	↑	(5)
19	:	8	=	2	|	(3)
2	:	8	=	0		(2)

Rješenje $157_{10}=235_8$

Konverzija oktalnih brojeva u dekadne je jednostavna:
$2\cdot 8^2+3\cdot 8^1+5\cdot 8^0 = 128+24+5=157$

1.6.3 Konverzija binarnih brojeva u oktalne i obrnuto

Pošto je $2^3 = 8$, znači za jedan jednocifren oktalni broj treba tri bita.
Prema tome, binarni brojevi se mogu podijeliti u grupe po tri bita,počevši od pozicionog zareza. Svakoj takvoj grupi može se pripisati jedan oktalni broj.
Zadatak: $(10010100100011111)_2 = (\ ?\ )_8$
Rješenje podjelimo binarni zapis u grupe po tri cifre i svakoj grupi pridružimo oktalnu vrijednost:

$$(10'\ 010'\ 100'\ 100'\ 011'\ 110)_2 = (224437)_8$$

Analogno oktalni broj se vrlo jednostavno pretvara u binarni.
Npr. ako želimo da **Primjer:** $(471)_8 \rightarrow (\ ?\ )_2$
Napravimo jednostavnu tabelu sa oktalnim ciframa i njima pripadajućim binarnim brojevima

	4 (2^2)	2(2^1)	1(2^0)
4	1	0	0
7	1	1	1
1	0	0	1

i na analogno predhodnom (grupisanju po tri cifre) dobiti da je:
$(471)_8 \rightarrow (100111001)_2$

1.6.4 Konverzija binarnih brojeva u heksadecimalne i obrnuto

Svaki binarni broj koji treba pretvoriti u heksadecimalni dijeli u grupe po četiri bita i svakoj grupi posebno dodjeljuje odgovarajući haksadecimalni ekvivalent.
Razlog je očigledan: osnova heksadecimalnog sistema 16 , što je 2^4 .
To pruža mogućnost da zapis u binarnom sistemu lakše pamtimo koristeći heksadecimalni zapis.

Primjer:
$(1011101000011111)_2 = 1011\ 1010\ 0001\ \ 1111$
$(\ B\quad A\quad 1\quad F\ )_{16}$
Dugi niz binarnih cifri se grupiše po četri i svaka grupa se konvertuje u odgovarajuću heksadecimalnu cifru, npr.:
$(11'1101'0010'0110'1010'0011)_2 = 0011\ \ 1101\ \ \ 0010\ \ \ 0110\ \ 1010\ \ 0011 = (3D26A3)_{16}$

Za povratak u binarni sistem, svakoj heksadecimalnoj cifri pripisuje se binarni ekvivalent sa četiri cifre.
Primjer konverzije heksadecimalnog u binarni:

$(F6B2)_{16}$ = F 6 B 2
1111 0110 1011 0010 = $(1110011010110010)_2$

1.6.5 Konverzija decimalnih u heksadecimalne i obrnuto

Konverzija koristi isti metod objašnjen kod binarnih brojeva, *samo je različita baza i cifre.*

Primjer (jednostavne) konverzije jednostavne dekadnog broja 28 u heksadecimalni broj.
Procedura dijeljenja:

$$\begin{array}{rcl} 28:16 & = & 1 \\ 1:16 & = & 0 \end{array}$$

Kreiranje heksadecimalne cifre:

$$P = 28_{10} = 1 \cdot 16^1 + 12 \cdot 16^0 = 1 \cdot 16^1 + C \cdot 16^0 = 1C_{16}$$

Primjer: $\mathbf{1067_{10}=?_{16}}$

1067	:	16	=	66 ($11_{10}=B_{16}$)	(B)
66	:	16	=	4 ($2_{10}=2_{16}$)	(2)
4	:	16	=	0 ($4_{10}=4_{16}$)	(4)

Rješenje $1067_{10}=\mathbf{42B_{16}}$

Obrnuta konverzija je očigledna i jednostavna:
$4\cdot 16^2+2\cdot 16^1+11\cdot 16^0 = 1024+32+11=1067$

2 Osnovne aritmetičke operacije u binarnom brojnom sistemu

Osnovne operacije su sabiranje, oduzimanje, množenje i dijeljenje. One se provode ekvivalentno kao što smo navikli u dekadnom brojnom sistemu sa pravilima, prenosa i pozajmice. „Jedina“ razlika je što binarni sistem ima samo dvije cifre.
Pravila koja važe za ove operacije mogu se predstaviti tablicama.

Upoznajte sa tablicama sabiranja, oduzimanja i množenja koje vrijede za brojeve koji pripadaju binarnom brojnom sistemu i onda ćemo kroz nekoliko primjera prezentovati osnovne tehnike sabiranja, oduzimanja, množenja i dijeljenja binarnih brojeva.

Tablica **sabiranja** (+)	Tablica **oduzimanja** (-)
0 + 0 = 0 1 + 0 = 1 0 + 1 = 1 1 + 1 = 0 (1 se prenosi)	0 - 0 = 0 1 - 0 = 1 **0 - 1 = 1 (1 se pozajmljuje)** 1 - 1 = 0
Tablica **množenja**	Za binarno **dijeljenje** važe pravila:
0 * 0 = 0 1 * 0 = 0 0 * 1 = 0 1 * 1 = 1	0 : 1 = 0 1 : 1 = 1, Dijeljenje sa nulom nije dozvoljeno.

Tabela 2 Tablice osnovnih aritmetičkih operacija

2.1 Primeri aritmetičkih operacija

Zadatak1. Izračunajte zbir binarnih brojeva: a) 10 i 11 b) 10 i 10
Rješenje:

10 11 **101**	Komentar: 1+0 jednako je 1 1+1 jednako je 0, a 1 se prenosi (″pamtite″ 1).
10 10 **100**	

Zadatak2. Izračunajte razliku binarnih brojeva: a) 1011 i 101 b) 1000 i 111
Rješenje:

1011 101 **110**	Komentar: 1-1 jednako je 0, 1-0 jednako je 1 0-1 jednako je 1, a 1 se pozajmljuje, 1-1(pozajmljeno)
1000 111 0001 **1**	0-1 jednako je 1, a 1 se pozajmljuje, 1-1(pozajmljeno)

Zadatak3. Izvrši sabiranje *SUM=A+B*= 11 1011 + 10 1010
Rješenje:

Prenos	1	1	1		1		
A		**1**	**1**	**1**	**0**	**1**	**1**
B	+	**1**	**0**	**1**	**0**	**1**	**0**
Sum	**1**	**1**	**0**	**0**	**1**	**0**	**1**
koraci	*7*	*6*	*5*	*4*	*3*	*2*	*1*

Pomoć:

Korak k	**$A_k+B_k+Prenos_k = Prenos_{k+1}Sum_k$**
1	1 + 0 = 1 sum 1
2	1 + 1 = 10 → prenos 1 sum 0
3	0 + 0 + 1 = 1 → sum 1
4	1 + 1 = 10 → prenos 1 sum 0
5	1 + 0 + 1 = 10 → prenos 1 sum 0
6	1 + 1 + 1 = 11 → prenos 1 sum 1
7	1 = 1 → sum 1

Prenose je zgodno zapisivati po „razredma“ IZNAD sabiraka da ih lakše pratimo

Zadatak 4. Izračunajte proizvod binarnih brojeva: 1011 i 11.
Rješenje:

1011*11 _____ 1011 1011 _____ 100001	Komentar: Množite po tablici množenja, a sabiranje vršite po tablici za sabiranje binarnih brojeva

Množenje binarnih brojeva se izvodi tako da izvršimo pomjeranje ulijevo i dopišemo onoliko nula kolika je težina pozicije cifre sa kojom množimo. (normalno ukoliko je ta cifra 1, u slučaju da je nula preskačemo).
Za pomjeranje se koristi termin „šiftovanje“(od Shift-pomjeri*), šiftovanje ulijevo znači pomjeranje svih bita broja ulijevo.* Množenje se svodi na šiftovanje ulijevo dopisivanje nula, uz „potpisivavanje sa pomjeranjem“. Poslije toga saberemo ovako „potpisane“ brojeve i dobijemo rezultat.

Primjer 1: pomnoži *1 1101 x 100*

A				1	1	1	0	1
B					×	1	0	0
Rezultat	1	1	0	0	1	1	0	0

Primjer 2: pomnoži *1 1101 x 111*

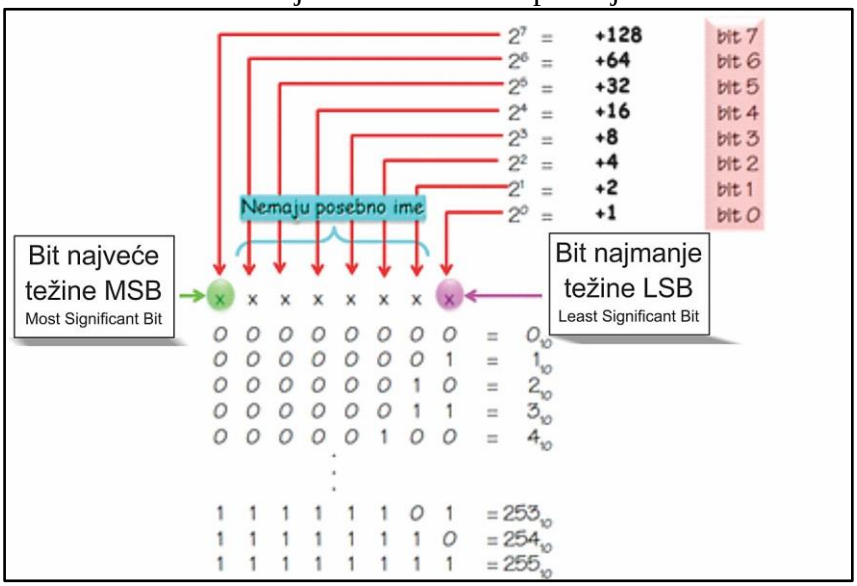
A				1	1	1	0	1
B					×	1	1	1
		1	1	1	0	1	0	0
			1	1	1	0	1	0
				1	1	1	0	1
Rezultat	1	1	0	0	1	0	1	1
Prenos	1	10	10	1	1			

3 Predstavljanje i označavanje binarnih brojeva

Termin označeni broj se odnosi na brojeve kod kojih postoji informacija o predznaku, odnosno o eventualnoj negativnoj vrijednosti broja. U decimalnom brojnom sistemu negativni brojevi se predstavljaju znakom "-" (pozitivni znakom "+" ili se znak izostavlja) napisanim ispred cifara koje definišu apsolutnu vrijednost broja. U binarnom brojnom sistemu ovakav način predstavljanja označenih brojeva nije moguć, jer na raspolaganju imamo samo dva znaka ("0" i "1").

3.1 Neoznačeni cijeli binarni brojevi

Neoznačeni (unsigned) binarni brojevi se koriste samo za prikaz pozitivnih brojnih vrijednosti. Na Slici 2 prikazan je opseg brojeva koji se mogu reprezentovati u binarnom brojnom sistemu sa 8 pozicija.



Slika 2

Na lijevoj strani se nalazi **bit najveće težine** - MSB (**Most Significant Bit**), koji se indeksira se kao bit 7.
Na krajnjoj desnoj strani se nalazi **bit najmanje težine** - LSB (**Least Significant Bit**), *koji se* indeksira se kao bit 0.
Svaki bit može uzeti vrijednost 1 ili 0, tako da grupa od 8 bitova može predstaviti 2^8 jedinstvenih kombinacija 0 i 1. što odgovara **opsegu od 0_{10} do $+255_{10}$** u decimalnom zapisu.

3.2 Označeni cijeli binarni brojevi sa bitom znaka

Označeni brojevi (engl. signed) **se koriste za predstavljanje pozitivnih i negativnih brojnih vrijednost**i. Za predstavljanje brojeva koji imaju negativnu vrijednost se kod standardne decimalne notacije koristi zapis u formi **znak – apsolutna vrijednost**, npr.: -10 i +10.

Umjesto znaka minus (-) **u računarima se koristi bit znaka** (engl. sign-bit) kao način označavanja negativnih brojeva. Kod označenih binarnih brojeva, bit najveće težine: MSB je bit znaka i određuje znak broja tako ako je:

- Bit znaka: MSB bit = 0, onda se radi o pozitivnom broju;
- Bit znaka: MSB bit =1, onda se radi o negativnom broju.

*Označavanje negativnih brojeva **korišćenjem bit znaka** poznato je i kao kodovanje negativnih brojeva korišćenjem direktnog koda.*

Daćemo nekoliko jednostavanih primjera i zadataka.

Primjer1:	
7_{10} = 111_2 $+7_{10} = 0111_2$ $-7_{10} = 1111_2$	neoznačen binarni broj označen pozitivan binarni broj označen negativan binarni broj
Primjer 2:	
3_{10} = 11_2 $+3_{10} = 0011_2$ $-3_{10} = 1011_2$	neoznačen binarni broj označen pozitivan binarni broj označen negativan binarni broj

Primjer 3 Odnos dužine riječi (broja bita)

8 bitni	**16 bitni**
0 1 1 1 1 1 1 1 …….127 0 1 1 1 1 1 1 0……..126 0 0 0 0 0 1 1 1……7 1 0 0 0 0 1 1 1……-7 11 1 1 1 1 1 1 ……-127	0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 1……29 1 0 0 0 0 0 0 0 0 0 1 0 1 1 1 0 …… - 46

Zadatak: Koji broj je zapisan u 8 bitnom registru ako je sadržaj registra **10010011**
prva jedinica → predznak je negativan: - ; broj 10011 pretvorimo u dekadni
$=1\cdot2^4+0\cdot2^3+0\cdot2^2+1\cdot2^1+1\cdot2^0=21$
Rješenje je **-21**

Pri radu sa binarnim brojevima zapisanim pomoću znaka i apsolutne vrijednosti javlja se **problem** pri obavljanju aritmetičkih operacija zato što se negativan broj ne tretira na jedinstven način. *Problem se rješava predstavljanjem negativnih binarnih brojeva u komplementu dvojke*.

3.3 Označavanje brojeva korišćenjem komplementa

Puni komplement ili **komplement dvojke** binarnog n bitnog broja dobija se kao dopuna do broja 2^n odnosno, vrijednost komplementa dvojke za neki broj dobija se **oduzimanjem tog broja od broja 2^n**. Efekat oduzimanja od 2^n može se postići i invertovanjem (**komplement jedinica**) tako da se sve cifre broja se **invertuj**u (jedinice se zamijene nulama, a nule jedinicama) i dodavanjem 1: Kad komplement jedinice saberemo sa 1 dobije se **puni komplement ili komplement dvojke.**

Ne-negativni broj (**pozitivni**) brojevi prikazuju se isto kao neoznačeni samo što MSB bit obavezno 0.
Opseg prikazanih brojeva nije isti kao kod neoznačenih (unsigned) binarnih brojeva. n-bitni broj predstavljen komplementom dvojke može da predstavlja pozitivne brojeve do 2^{n-1}. Tako 8-bitni broj komplementa dvojke može da predstavlja cijele pozitivne brojeve od 0 do 127 (01111111). Ostatak kombinacija bitova sa MSB=1 predstavlja cijele negativne brojeve od -1 do -128.
Operacija komplementa dvojke je **aditivno inverzna operacija**, pa su *negativni brojevi predstavljeni drugim komplementom apsolutne vrednosti*.

Negativni brojevi mogu da se označe i koduju i korišćenjem komplementa dvojke. Negativan označeni broj se dobija tako da se:

1. ispred neoznačenog binarnog broja doda cifra 0
2. sve cifre broja se komplementiraju (**kompliment jedinice)**
3. komplement jedinice sabere sa 1 dobije se **puni komplement ili komplement dvojke koji predstavlja negativan broj.**

Tabela 3 Označavanje brojeva

decimalni sa znakom	binarni sa znak bitom	binarni u komplementu dvojke
-8	-	1000
-7	1111	1001
-6	1110	1010
-5	1101	1011
-4	1100	1100
-3	1011	1101
-2	1010	1110
-1	1001	1111
0	1000 ili 0000	0000
1	0001	0001
2	0010	0010
3	0011	0011
4	0100	0100
5	0101	0101
6	0110	0110
7	0111	0111

Sabiranje i oduzimanje brojeva predstavljenih komplementom dvojke je mnogo jednostavnije nego kod brojeva predstavljenih za bit znakom. Postoji sam jedna predstava broja 0 (*čime je eliminisan problem dvostrukog tretmana nule, u slučaju označavanja bit znakom, ili nepotpunom komplementu*).

Obim označenih brojeva predstavljenih u komplimentu dvojke je:
$x \leq 2^{n-1}$ za $x \geq 0$,
$|x| \leq 2^{n-1}$ za $x<0 \Rightarrow x \in \{-2^{n-1},\ldots -1,0,1,\ldots 2^{n-1}-1\}$
gdje je n broj cifara.

Zadatak: Decimalni broj -6_{10} predstaviti u komplementu dvojke
Rješenje:
Ispred neoznačenog broja 6=110 dodaje se 0 i dobije se 0110
Komplement jedinice 1001
Komplement dvojke 1001+1 = 1010
Konačni rezultat (komplement dvojke): $\mathbf{-6_{10}=1010}$

3.4 Sabiranje i oduzimanje brojeva zapisanih u komplementu dvojke

Ako se umanjilac kodira u komplementu dvojke onda se operacija oduzimanja može provesti tako što se tako kodovan umanjilac sabere sa umanjenikom.
Oduzimanje se svodi na sabiranje, po formuli: **A-B=A+(-B)** (odnosno koristimo reprezentaciju punog komplementa za sabiranje koje daje rezultat oduzimanja).
Osobina komplementa dvojke: kada se dva puta uzastopno primjeni na neki broj, dobija se polazni broj → Oduzimanje negativnog brojeva odgovara sabiranju pozitivnog.

I Sabiranje i oduzimanje obavlja se prema pravilima sabiranja tako da se poštuje sljedeća procedura:

1. Pozitivni brojevi se označe bit znakom: ispred vrijednosti se upiše nula MSB bit kako je ranije opisano.
2. Negativni brojevi se predstave u komplementu dvojke.
3. Ukoliko nemaju isti broj cifara, sabirak sa manjim brojem cifara se dopuni **vodećim nulama ako je pozitivan** ili **jedinicama ako je negativan** do broja cifara drugog sabirka
4. **Tako dobijeni brojevi se saberu po pravilima binarnog sabiranja ne vodeći računa o znaku**
5. Dobijeni rezultat (uzima se isti broj cifara kao kod sabiraka) je također u komplementu dvojke

Primjer svođenja oduzimanja na sabiranje sa punim komlementom:

a)

```
   10                01010        (I)
 - 6               - 00110
   4
```

Pošto se radi o oduzimanju umanjioc treba prebaciti u puni komplement invertovanjem → 00110 → 11001 i sabiranjem sa 1

```
 11001
+    1
 11010        (II)
```

Tek sada se prelazi na sabiranje (I) i (II)

```
  01010
+ 11010
  00100
```

Kako je najznačajniji bit 0 to je rezultat pozitivan i ostaje onakav kakav jeste.

b) Koristeći drugi kompliment izračunaj 7_{10}-11_{10}
Binarno 00111_2 - 01011_2
Prebaci drugi broj u komplement dvojke
Inverovanje → 01011 → 10100 i sabiranjem sa jedan

```
   10100
 +     1
-------
   10101      (puni komplement)
```

Sabiranje:

```
   00111
 + 10101
-------
   11100
```

Kako je najznačajniji bit 1 to je rezultat negativan i moramo ga ponovo konvertovati u puni komplement: 11100→ 00011

```
     +    1
     -------
     00100 => (- 4)10
```

Posljedica ovakvog pristupa (izvođenja operacije oduzimanja korišćenjem dvojnog komplementa i sabiranja) je da se sabiranje koristi jedinstven hardverski dio „sabirač“. pomoću koga se mogu izvršiti sve aritmetičke operacije (oduzimanje kao komplementiranje i sabiranje, množenje kao višestruko sabiranje a djeljenje kao višestruko oduzmanje).

Napomena: Rezultat sabiranja mora biti **u dozvoljenom opsegu** brojeva koji se mogu predstaviti datim brojem cifara **da ne bi došlo do prekoračenja i greš**ke prilikom izračunavanja.

3.5 Predstavljanje realnih brojeva sa fiksnom decimalnom tačkom

Ako broj ima unaprijed definisanu i zadanu pozicija decimalne tačke riječ je o predstavljanju brojeva sa fiksnim decimalnim zarezom[1] (fiksnom decimalnom tačkom). Fiksira se broj pozicija za cjelobrojni dio, odnosno za razlomljeni dio. Za znak broja se rezerviše MSB bit (najteži/najveći) bit u cjelobrojnom dijelu tako da u opštem slučaju ima m n bitova za razlomljeni (fraktal) dio i m bitova za cijeli dio.

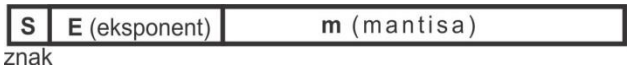
Ilustracija 3 Predstavljanje brojeva sa fiksnom decimalnom tačkom

Kod brojeva sa fiksnom decimalnom tačkom se unaprijed zna broj pozicija za razlomljeni dio. Broj pozicija (bitova) za razlomljeni dio zavisi od konkretne izvedbe računara. U našem prijemjeru na Slici 3 je m+n+1= 16 , za razlomljeni dio predviđeno je 8, a za cio dio 7 bita.

3.6 Predstavljanje brojeva sa pokretnom decimalnom tačkom

Predstavljanje razlomljenih (fraktalnih) brojeva sa fiksnom pozicijom decimalne tačke je neefikasno za prikazivanje veoma velikih i veoma malih brojeva.
Svaki razlomljeni broj R moguće je predstaviti sa $R=mb^E$, gdje je m mantisa broja R, E odgovarajući eksponent, a b baza brojnog sistema kao na Slici 4.

Ovakav format predstavljanja brojeva naziva se predstavljanje u pokretnom zarezu (**Float pointing, FP**), jer je pozicija zareza određena vrijednošću eksponenta.



Slika 4 Predstavljanje brojeva u pokretnom zarezu

[1] uobičajeno se koristi anglosaksonska oznaka tačka umjesto kod nas standardnog zareza

Normalizovani i nermalizovani zapis brojeva u pokretnom zarezu

Jedan od nedostataka predstavljanja brojeva u pokretnom zarezu je što se brojevi mogu predstaviti na nekoliko načina, tj. ne postoji jedinstvena predstava za jedan broj.

Tako se broj 25.125 može predstaviti kao 0.0025125×10^{3} ili kao 2512.5×10^{-2}.

Ako na ovaj način vršimo predstavljanje brojeva kažemo da su oni nenormalizovani. Normalizovani su oni brojevi kod kojih mantisa zadovoljava uslov:

$1 > F \geq 1/R$

Tako bi predhodni broj predstavljen u normalizovanoj formi broj 37.145, za bazu 10 bio: 0.25125E+2.

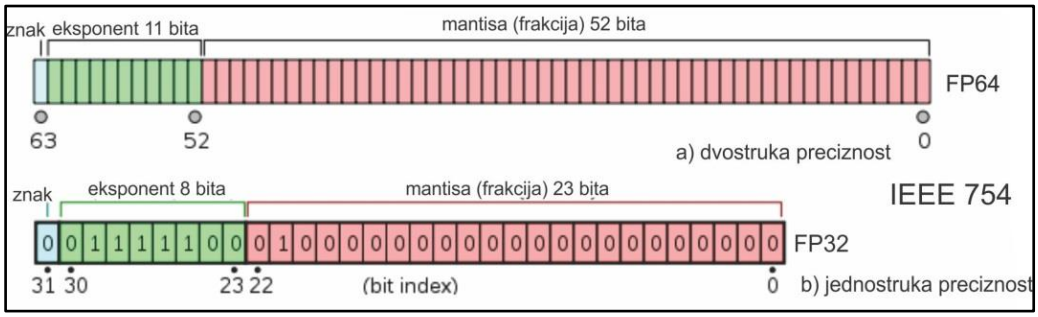
Međutim normalizovana notacija ne objezbeđuje predstavljanje određenih brojeva blizu nule. To znači da ako želimo predstaviti brojeve manje od najmanjeg normalizovanog (vidi Tabelu 2.5) koristićemo denormalizovane brojeve.

Pri radu sa pokretnim zarezom definisani su posebni ekstremni slučajevi, kao što je mogući rezultat NaN, *Not a Number*.

NaN su specijalne vrednosti koje nisu brojevi i označavaju neke izuzetne situacije prilikom izračunavanja.

Npr. 0/0 ili √-1 Eksponent NaN vrednosti je maksimalan. U slučaju jednostruke tačnosti to je 128 (11111111). Frakcija mora biti različita od nule.

Korektna analiza ovih slučajeva prevazilazi naše trenutne potrebe, a za njeno provođenje preporučujemo neki napredniji priručnik iz digitalne tehnike.



Slika 5

U računarima se koristi baza 2, pa je broj R moguće je predstaviti sa $R=m2^{E}$.

Način upotrebe brojeva u pokretnom zarezu je standardizovan formatom IEEE 754. Ovaj standard propisuje dva formata za rad u pokretnom zarezu i prikazan je na Slici 4.

Ukoliko se za predstavljanje brojeva koristi 32 bitni zapis takav format se naziva **jednostruka tačnost**-preciznost (***single precision***, *single*).

Za eksponent (E) se koristi 8 bita i može biti pozitivan i negativan tj. $e=E+127_{10}$.

Jedan bit (S, sign) se koristi za predznak.

Mantisa ima format $m=\overline{1.f}$
gdje se sa f predstavljaju preostale (23) binarne cifre. Na ovaj način se broj R može predstaviti uređenom trojkom R(s,e,f).

Normalizovani brojevi u jednostrukoj tačnosti imaju eksponent između -126 (00000001) i +127 (11111110). Frakcija ima oblik 1, d-1…d-(p-1), pri čemu se prva jedinica ne zapisuje. Zapis frakcije može da ima bilo koju vrijednost.

Nevidljivi bit
Ako su brojevi normalizovani najznačajniji bit mantise (MS) će uvijek biti 1 (osim za broj 0), što znači da ne nosi nikakvu informaciju. To omogućava da se mantisi dodjeli još jedan dodatni bit. Kad se vrši izračunavanje treba ga uzeti u obzir, a kada se upisuje rezultat, njega ne treba upisivati. Ovaj bit je poznat kao nevidljivi/skriveni (**hidden bit**) i značajan je kod interne konverzije podatka unutar procesora.

Predstavljanje nule u pokretnom zarezu može biti problem. Standardno se u slučaju da mantisa ima na svim cifarskim pozicijama nulu broj se interpretira kao nula.

Označena nula
Nula se u jednostrukoj tačnosti predstavlja eksponentom -127 (00000000) i frakcijom 0.
Pošto znak može biti + ili – (0 ili 1) onda postoje i dvije nule +0 i -0.
Prema standardu važi +0 = -0.
log-0 = NaN, a log+0 = -∞

Ukoliko je dužina cijelog broja 64 bita tad je riječ o **dvostrukoj preciznosti**. Rad sa brojevima obavlja se kao i u jednostrukoj precisnosti, samo što je proširen opseg, kao na Slici 5.b): eksponent ima 11, a frakcija 52 bita.

U Tabeli 4 date su najznačajnije karakteristike koje određuju rad sa pokretnim zarezom u jednostrukoj i dvostrukoj preciznosti.

stavka	**jednostruka preciznost**	**dvostruka preciznost**
bit znaka	1	1
bitovi eksponenta	8	11
bitovi mantise	23	52
ukupno bitova	32	64
sistem eksponenta	modifikovan za 127	modifikovan za 1023
opseg eksponenta	-126 do + 127	-1022 do + 1023
najmanji normalizovani	2^{-126}	2^{-1022}
najveći normalizovani	aproksimativno 2+128	aproksimativno 2+1024
decimalni opseg	aproksimativno od 10^{-38} do 10^{+38}	aproksimativno od 10^{-308} do 10^{+308}
najmanji denormalizovani	aproksimativno 10^{-45}	aproksimativno 10^{-324}

Tabela 4 Osnovne karkteristike IEEE FP standarda

Primjer 1
Prikazati broj 5.75 kao realni broj
Rješenje korak po korak:

1. Konvertovati dekadni broj u binarni: $5.75_{10} = 101.11_2$
2. Odrediti predznak: broj je pozitivan → S = 0
3. Izvršiti normalizaciju binarnog broja → $101.11_2 \cdot 2^0 = 1.\mathbf{0111}_2 \cdot 2^2$
4. Izračunati eksponent i predstaviti ga binarno

E = $2_{10} + 127_{10} = 129_{10} = 1000\ 0001_2$

5. Izbaciti vodeću jedinicu iz mantise (nevidljivi/skriveni bit)

M (bez nevidljivog bita i binarne tačke) = 0111_2

6. Prepisati predznak, karakteristiku i mantisu bez nevidljivog bita u registar

0	10000001	01110000000000000000000
S	eksponent	mantisa bez nevidljivog bita

$0100\ 0000\ 1011\ 1000\ 0000\ 0000\ 0000\ 0000_2$
$4\quad 0\quad B\quad 8\quad 0\quad 0\quad 0\quad 0_{16}$

Primjer 2
Zapisati broj 13,25 prema standardu IEEE 754 u jednostrukoj tačnosti.

Rješenje:
$(13,25)_{10} = (1101,01)_2 = (1,10101)_2 \cdot 2^3$

Predznak → S = 0
Eksponent sa uvećanjem 127:
$130 \rightarrow (10000010)_2$

Značajni dio bez nevidljivog bita → 10101 - $(1,10101)_2$

Zapis: 0 10000010 10101000000000000000000

4 Korišćenje kodova za zapis podataka

Ovdje se pod kodovanjem smatra prevođenje podataka u znakove s ciljem da se omogući njihova obrada putem računara: *Kodovanje je uspostavljanje odnosa između elemenata nekog skupa i riječi nekog alfabeta (koji ne mora nužno biti jednoznačan).*

4.1 Kodovi za zapis brojeva

Binarni brojni sistemi odgovaraju težinskim kodovima, jer se kod njih svakom binarnom simbolu se pripisuje određena težina.
Postoje i kodovi gdje binarne cifre nemaju odgovarajuću težinu, već je bitan samo njihov redoslijed u kodnoj riječi. Takvi kodovi se nazivaju redosljedni.

Binarno kodovani dekadni brojevi se koriste radi jednostavnijeg i tačnog zapisa dekadnih brojeva u računarskom sistemu. Princip zapisa je da **se svaka dekadna cifra koduje određenim binarnim zapisom**.

Pitanje: Koliko kodova je moguće napraviti ako dekadne cifre (0-9) želimo predstaviti sa četiri binarne cifre (0000-1111)?

4.2 BCD kod

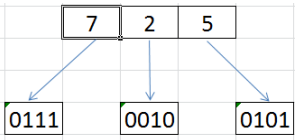
Binarno kodovani decimalni kod: BCD kod (**Binary Coded Decimal**) je posredni kod između čistog binarnog i decimalnog koda. Realizovan je kao niz ćelija, od kojih svaka ima četiri bita (jednu tetradu) i predstavlja jednu decimalnu cifru. U odnosu na standardnu binarnu prezentaciju kod BCD kodovanja svaku decimalnu cifru kodujemo binarno.

Decimal	0	1	2	3	4	5	6	7	8	9
BCD	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

Ilustracija 6 BCD kod

Ostale kombinacije (moguće u binarnom i heksa zapisu A-F) su neiskorišćene i ***predstavljaju nelegalna stanja.***

Tako decimalni broj 725 kod standardne binarne prezentacije oblika je 1111 0111, a u odgovarajućem BCD kodu ima oblik: 0111 0010 0101
Sve tri cifre su kodovane binarno.



Ilustracija 7
Primjer BCD koda

Ulazni i izlazni podaci iz računarskog sistema formiraju se u decimalnom obliku, ali se u računaru pomoću ovog koda svaka decimalna cifra predstavlja sa 4 bita tzv. tetrade, koje predstavljaju broj u binarnom obliku. Ovo olakšava rad čovjeku (jer je on najčešće u kontaktu samo sa ulaznim i izlaznim podacima), ali i računaru (koji jedino razumiije binarni zapis).

BCD kod je najčešće korišćeni težinski kod.

4.2.1 Sabiranje brojeva zapisanih u BCD kodu

Realizacija aritmetičkih operacija, kad se primjenjuje BCD kod, je složenija nego kad se radi sa brojevima koji su direktno binarno zapisani.
Kod BCD 8421, koji se još zove i prirodni BCD kod (NBCD), jer uzima prvih deset kombinacija od 4 bita za kodovanje decimalnih simbola (po težinskom principu 8421, kao i kod prirodnih binarnih brojeva). Njegova je prednost da može koristiti normalnu binarnu tehniku pa se često koristi u digitalnim uređajima.

Konverzija decimalnog sistema u kod 8421 je veoma jednostavna. Za svaku decimalnu cifru nađe se četverocifreni binarni ekvivalent.

Npr. Konverzija decimalnog broja 205:

2	0	5
0010	0000	0101

Za predstavljanje decimalnog broja ovaj kod očito zahtijeva više cifarskih mjesta nego prirodni binarni kod.

Proces sabiranja BCD brojeva je isti kao i kod binarnih brojeva sve dok je decimalna suma 9 ili manja.

decimalni		BCD	
5		0101	
+ 4		+ 0100	
9		1001	

Nedostatak BCD 8421 koda je što se na njega ne mogu uvijek primijeniti pravila binarnog sabiranja.

U slučaju da je rezultat sabiranja nelegalno stanje korekcija se vrši tako da se doda broj $6_{10}=0110_{BCD}$ koji vrši korekciju tako što se u prvoj tetradi dobije ispravan kod,a u drugu izvrši prenos.

Ilustrujmo to na dva primjera sabiranja:

decimalni	BCD		decimalni	BCD
8	1000		8	1000
+ 7	+ 0111		+ 8	+ 1000
15	1111	(15 nelegalan BCD kod)	16	0001 0000
	+ 0110	dodaje se 6		
	$0001\ 0101_{BCD} \rightarrow 15_{10}$			10000_{BCD}

4.3 Kodovi za zapis teksta kod računara

Kodiranje je način predstavljanja informacija pomoću simbola. Različiti simboli se u računaru kodiraju različitim varijacijama sa ponavljanjem jedinica i nula fiksne dužine u kodu fiksne dužine. Kodne riječi u jednom kodu fiksne dužine nazivaju se karakteri (characters).
Tekst se u računarima predstavlja kao niz karaktera koji pripadaju određenom alfabetu. Pri zapisu teksta koristi se skup znakova koji sadrži slova, znakove interpunkcije, neke komercijalne simbole i aritmetičke operatore. Kao posebna klasu može se izdvojiti skup znakova koji služe kao upravljački znaci (control characters) to su specijalni karakteri koji označavaju prelazak u novi red, tabulator, kraj teksta i slično.

Zapis teksta na računaru se obavlja tako da se svakom karakteru pridruži određeni (neoznačeni) cio broj na unapred dogovoreni način. Ovi brojevi se nazivaju kodovima karaktera (character codes).

Kodna šema definiše značenje znaka (karaktera) i njegovu binarnu reprezenta-ciju, predstavlja standard koji određuje značenje znaka i njegov stvarni zapis (binarnu reprezentaciju). **Karakter je određen spoljašnjim oblikom** (grafički oblik, koji se prepoznaje kao znak u određenom sistemu pisanja; slovo, cifra, znak interpunkcije ili specijalni znak) **i svojom unutrašnjom reprezentacijom: kodnom šemom**.

S obzirom na broj bitova potrebnih za kodiranje odredenih karaktera, razlikuju se 7-bitni kodovi, 8-bitni kodovi, 16-bitni kodovi, 32-bitni kodovi, kao i kodiranja promjenljive dužine koja različitim karakterima dodeljuju kodove različite dužine.

Tokom godina razvijeni su različiti standardi kodovanja. 5 najčešće korišćenih standarda (kodnih šema) koji definišu kodove kod računara su:

1. ASCII 7-bitni kod, 128 karaktera, i prošireni ASCII-8bitni.
2. EBCDIC 8-bitni kod, 256 karaktera.
3. ISO-8 . 8-bitni kod, 256 karaktera.
4. IBM-PC kod. 8-bitni kod, 256 karaktera
5. UNICODE (UNIversal encODE). 16-bitni kod, 65536 karaktera

U praksi su najzastupljeniji prvi (ASCII) i posljednji (Unicode), pa se ostalima nećemo ni baviti.

4.3.1 ASCII-ošišana latinica2

ASCII („Američki standardni kod za razmjenu podataka“ engl. American Standard Code for Information Interchange, izgovara se „aski“) je standard koji definiše osnovni skup (niz) brojeva potreban za definisanje znakova i slova za unos teksta.

ASCII kod omogućava da na računaru pišemo tekst koji ima sve znakove i slova pod uslovom da govorimo engleski.

Prvi računari pravljeni su za englesko govorno područje i imali su podršku samo za engleski alfabet, za brojeve, zagrade i neke kontrolne karaktere što je činilo ukupno 128 mogućih slova.

Koliko nam za to treba bita?
U prethodnom poglavlju je objašnjeno (obim koda) zašto je dovoljno 7 bita.

Korisnicima i programerima se dostavljala neka tabela (slična Tabeli 6) i oni su mogli da pristupe kodovanju.
Kao standard ASCII je prvi put objavljen 1967., a dorađen je 1986.
U novoj verziji definisani su kodovi za 33 kontrolna znaka, kojima se utiče na način ispisa teksta (koji se ne štampaju), te sljedećih 95 znakova za sam prikaz teksta; prvi znak je razmak. U ASCII tabeli to su kontrolni karakteri do koda 32, a i kod 127.

[2] na našim prostorima za ovaj kod se koristi i ovaj naziv, jer ako njega koristimo slovima latinice nedostaju ona sa apokrifima-kvačicama (nedostaju kvačice na slovima š, č, ć, ž i đ.)

Prošireni set je zadržao stari naziv ASCII sa prefiksom prošireni, koji se često izostavlja i uzrokuje zabunu. (možete ga vidjeti u Tabeli 7)
Za prošireni ASCII kod potrebno je 8 bita. (Zapravo zajedno za prošireni i osnovni).

ASCII je ugrađen u unikod, kao prvih 128 kombinacija.
Mada je jasno samo po sebi, daćemo dva primjera kodovanja i načina čitanja koda iz tabele: Slovo A ima kod 65_{10}, ili 41_h ili 0100000_b
znak + ima kod 43_{10} ili $2B_h$ ili 00101010_b

Dec	Hex	Name	Char	Ctrl-char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	0	Null	NUL	CTRL-@	32	20	Space	64	40	@	96	60	`
1	1	Start of heading	SOH	CTRL-A	33	21	!	65	41	A	97	61	a
2	2	Start of text	STX	CTRL-B	34	22	"	66	42	B	98	62	b
3	3	End of text	ETX	CTRL-C	35	23	#	67	43	C	99	63	c
4	4	End of xmit	EOT	CTRL-D	36	24	$	68	44	D	100	64	d
5	5	Enquiry	ENQ	CTRL-E	37	25	%	69	45	E	101	65	e
6	6	Acknowledge	ACK	CTRL-F	38	26	&	70	46	F	102	66	f
7	7	Bell	BEL	CTRL-G	39	27	'	71	47	G	103	67	g
8	8	Backspace	BS	CTRL-H	40	28	(	72	48	H	104	68	h
9	9	Horizontal tab	HT	CTRL-I	41	29	)	73	49	I	105	69	i
10	0A	Line feed	LF	CTRL-J	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	VT	CTRL-K	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	FF	CTRL-L	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage feed	CR	CTRL-M	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	SO	CTRL-N	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	SI	CTRL-O	47	2F	/	79	4F	O	111	6F	o
16	10	Data line escape	DLE	CTRL-P	48	30	0	80	50	P	112	70	p
17	11	Device control 1	DC1	CTRL-Q	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	DC2	CTRL-R	50	32	2	82	52	R	114	72	r
19	13	Device control 3	DC3	CTRL-S	51	33	3	83	53	S	115	73	s
20	14	Device control 4	DC4	CTRL-T	52	34	4	84	54	T	116	74	t
21	15	Neg acknowledge	NAK	CTRL-U	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	SYN	CTRL-V	54	36	6	86	56	V	118	76	v
23	17	End of xmit block	ETB	CTRL-W	55	37	7	87	57	W	119	77	w
24	18	Cancel	CAN	CTRL-X	56	38	8	88	58	X	120	78	x
25	19	End of medium	EM	CTRL-Y	57	39	9	89	59	Y	121	79	y
26	1A	Substitute	SUB	CTRL-Z	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	ESC	CTRL-[	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	FS	CTRL-\	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	GS	CTRL-]	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	RS	CTRL-^	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	US	CTRL-_	63	3F	?	95	5F	_	127	7F	DEL

Tabela 5 ASCII standard

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
128	80	Ç	160	A0	á	192	C0	└	224	E0	α
129	81	ü	161	A1	í	193	C1	┴	225	E1	ß
130	82	é	162	A2	ó	194	C2	┬	226	E2	Γ
131	83	â	163	A3	ú	195	C3	├	227	E3	π
132	84	ä	164	A4	ñ	196	C4	─	228	E4	Σ
133	85	à	165	A5	Ñ	197	C5	┼	229	E5	σ
134	86	å	166	A6	ª	198	C6	╞	230	E6	µ
135	87	ç	167	A7	º	199	C7	╟	231	E7	τ
136	88	ê	168	A8	¿	200	C8	╚	232	E8	Φ
137	89	ë	169	A9	⌐	201	C9	╔	233	E9	Θ
138	8A	è	170	AA	¬	202	CA	╩	234	EA	Ω
139	8B	ï	171	AB	½	203	CB	╦	235	EB	δ
140	8C	î	172	AC	¼	204	CC	╠	236	EC	∞
141	8D	ì	173	AD	¡	205	CD	═	237	ED	φ
142	8E	Ä	174	AE	«	206	CE	╬	238	EE	ε
143	8F	Å	175	AF	»	207	CF	╧	239	EF	∩
144	90	É	176	B0	▒	208	D0	╨	240	F0	≡
145	91	æ	177	B1	▓	209	D1	╤	241	F1	±
146	92	Æ	178	B2	▓	210	D2	╥	242	F2	≥
147	93	ô	179	B3	│	211	D3	╙	243	F3	≤
148	94	ö	180	B4	┤	212	D4	Ö	244	F4	⌠
149	95	ò	181	B5	╡	213	D5	╒	245	F5	⌡
150	96	û	182	B6	╢	214	D6	╓	246	F6	÷
151	97	ù	183	B7	╖	215	D7	╫	247	F7	≈
152	98	ÿ	184	B8	╕	216	D8	╪	248	F8	°
153	99	Ö	185	B9	╣	217	D9	┘	249	F9	∙
154	9A	Ü	186	BA	║	218	DA	┌	250	FA	·
155	9B	¢	187	BB	╗	219	DB	▉	251	FB	√
156	9C	£	188	BC	╝	220	DC	▄	252	FC	ⁿ
157	9D	¥	189	BD	╜	221	DD	▌	253	FD	²
158	9E	Pts	190	BE	╛	222	DE	▐	254	FE	■
159	9F	ƒ	191	BF	┐	223	DF	▀	255	FF	

Tabela 6 ASCII prošireni kod

4.3.2 Kodne strane

Kako se računari upotrebljavaju i van engleskog govornog područja, treba se omogućiti i podrška za druge jezike. Podrška za *druge jezike* je omogućena upotrebom kodnih strana (Code Page).

Koncept je jednostavan. Zadržan je osnovni ASCII set a u proširenom su definisani znakovi za određeni jezik ili skupinu srodnih jezika.

Tako postoje

- Latin1 (ISO-8859-1) za latinična pisma Zapadne Evrope (Francuska, Njemačka, Španija,...),
- Latin2 (ISO-8859-2) i
- Windows-1250 za latinična pisma Istočne Evrope (gdje je smještena i naša latinica),
- ISO-8859-5, KOI8-R
- Windows-1251 za ćirilicu (vidi Tabelu 8)
- ...

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00	NUL 0000	STX 0001	SOT 0002	ETX 0003	EOT 0004	ENQ 0005	ACK 0006	BEL 0007	BS 0008	HT 0009	LF 000A	VT 000B	FF 000C	CR 000D	SO 000E	SI 000F
10	DLE 0010	DC1 0011	DC2 0012	DC3 0013	DC4 0014	NAK 0015	SYN 0016	ETB 0017	CAN 0018	EM 0019	SUB 001A	ESC 001B	FS 001C	GS 001D	RS 001E	US 001F
20	SP 0020	! 0021	" 0022	# 0023	$ 0024	% 0025	& 0026	' 0027	(0028	) 0029	* 002A	+ 002B	, 002C	- 002D	. 002E	/ 002F
30	0 0030	1 0031	2 0032	3 0033	4 0034	5 0035	6 0036	7 0037	8 0038	9 0039	: 003A	; 003B	< 003C	= 003D	> 003E	? 003F
40	@ 0040	A 0041	B 0042	C 0043	D 0044	E 0045	F 0046	G 0047	H 0048	I 0049	J 004A	K 004B	L 004C	M 004D	N 004E	O 004F
50	P 0050	Q 0051	R 0052	S 0053	T 0054	U 0055	V 0056	W 0057	X 0058	Y 0059	Z 005A	[005B	\ 005C	] 005D	^ 005E	_ 005F
60	` 0060	a 0061	b 0062	c 0063	d 0064	e 0065	f 0066	g 0067	h 0068	i 0069	j 006A	k 006B	l 006C	m 006D	n 006E	o 006F
70	p 0070	q 0071	r 0072	s 0073	t 0074	u 0075	v 0076	w 0077	x 0078	y 0079	z 007A	{ 007B	| 007C	} 007D	~ 007E	DEL 007F
80	Ђ 0402	Ѓ 0403	‚ 201A	ѓ 0453	„ 201E	… 2026	† 2020	‡ 2021	€ 20AC	‰ 2030	Љ 0409	‹ 2039	Њ 040A	Ќ 040C	Ћ 040B	Џ 040F
90	ђ 0452	‘ 2018	’ 2019	“ 201C	” 201D	• 2022	– 2013	— 2014		™ 2122	љ 0459	› 203A	њ 045A	ќ 045C	ћ 045B	џ 045F
A0	NBSP 00A0	Ў 040E	ў 045E	Ј 0408	¤ 00A4	Ґ 0490	¦ 00A6	§ 00A7	Ё 0401	© 00A9	Є 0404	« 00AB	¬ 00AC	- 00AD	® 00AE	Ї 0407
B0	° 00B0	± 00B1	І 0406	і 0456	ґ 0491	µ 00B5	¶ 00B6	· 00B7	ё 0451	№ 2116	є 0454	» 00BB	ј 0458	Ѕ 0405	ѕ 0455	ї 0457
C0	А 0410	Б 0411	В 0412	Г 0413	Д 0414	Е 0415	Ж 0416	З 0417	И 0418	Й 0419	К 041A	Л 041B	М 041C	Н 041D	О 041E	П 041F
D0	Р 0420	С 0421	Т 0422	У 0423	Ф 0424	Х 0425	Ц 0426	Ч 0427	Ш 0428	Щ 0429	Ъ 042A	Ы 042B	Ь 042C	Э 042D	Ю 042E	Я 042F
E0	а 0430	б 0431	в 0432	г 0433	д 0434	е 0435	ж 0436	з 0437	и 0438	й 0439	к 043A	л 043B	м 043C	н 043D	о 043E	п 043F
F0	р 0440	с 0441	т 0442	у 0443	ф 0444	х 0445	ц 0446	ч 0447	ш 0448	щ 0449	ъ 044A	ы 044B	ь 044C	э 044D	ю 044E	я 044F

Tabela 7 Windows 1251 kod

Problem sa kodnim stranama je što se međusobno isključuju. To je osnovni nedostatak ovog koncepta, jer nije dopuštao mogućnosti rada sa više pisama istovremeno, pa je cijeli dokument morao da bude pisan istim pismom.
Postoje rješenja koja dijelomično rješavaju mogućnost istovremene upotrebe više pisama, ali je tek sistem Unicode dao jedinstveno rješenje za sve jezike.

4.3.3 Unikod (unicode)

Unikod koristi jedinstven broj za svaki znak, bez obzira na program, jezik, platformu (operativni sistem). Unikod svaki znak predstavlja jednom kodnom tačkom, gdje je kodna tačka jednostavno cijeli broj.
Postoji više verzija Unikoda. Osnovna verzija UCS-2 koristi dvobajtni format zapisa što znači da je moguće kodovati $2^{16} = 65536$ različitih znakova.
Sa svojih 65536 karaktera Unicod riješava problem zapisa skoro svih postojećih pisama (uključujući čak i neka izmišljena, kao na primer klingonsko pismo).

Postoji i novi Unikod standard UCS-4 koji koristi četiri bajta za zapis.
UCS-4 je prevelik (zahtijeva previše prostora) i nepraktičan jer ima previše nula. Naročito je kritičan problem kontrolnih signala i kodovanja sa UCS-4 prilikom prenosa signala na internetu, ali i kod unix platformi.

Kodne tačke Unicode standarda se mogu podijeliti na različite načine, najčešće za 17 ravnina, svaka s 65 536 (= 2^{16}) kodnih tačaka. Trenutno se koristi tek nekoliko njih (oko 10% prostora):

- Plane 0 (0000–FFFF): **Basic Multilingual Plane (BMP). Osnovna ravnina**, u kojoj se nalazi većina znakova.
- Plane 1 (10000–1FFFF): Supplementary Multilingual Plane (SMP).
- Plane 2 (20000–2FFFF): Supplementary Ideographic Plane (SIP)
- **Planes 3 to 13 (30000–DFFFF) se za sada ne koriste**
- Plane 14 (E0000–EFFFF): Supplementary Special-purpose Plane (SSP)
- Plane 15 (F0000–FFFFF) rezervisano za privatnu upotrebu, Private Use Area (PUA)
- Plane 16 (100000–10FFFF), rezervisano za privatnu upotrebu, Private Use Area (PUA)

Prva ravnina (ravnina 0), tzv. Basic Multilingual Plane (BMP), sadrži većinu znakova kojima su do sada dodijeljene kodne tačke, i uključuje veliku većinu znakova koji se koriste u modernim jezicima. Većina kodnih tačaka se koristi za kodiranje kineskih, japanskih i korejskih znakova.

Postojao je problem alokacije prostora za Unicode poruku na medijumu koji se koristi.
Ako je riječ o nekom dokumentu na disku, on će da zauzima duplo više prostora nego konvencionalan dokument jer će se svaki karakter zapisivati sa dva bajta umesto samo sa jednim.
Ako je riječ o prenosu podataka preko računarske mreže, biće potrebno prenijeti duplo više podataka, pa će samim tim i prenos da traje duplo više (odnosno da košta duplo više).

Postavilo se pitanje da li je to suviše velika cijena za univerzalno pismo i da li postoji neki način da se taj problem prevaziđe i izbjegne. Kao rješenje uvijek stoji mogućnost da se zapisuje nekom odgovarajućom kodnom stranicom i troši bajt po karakteru, ako nije neophodno korištenje više pisama u istom dokumentu (što se rijetko dešava). Drugo rješenje je korištenje tzv. transformacionih šema za pogodniji zapis i prenos podataka korištenjem Unicode-a.

4.3.4 UTF-8

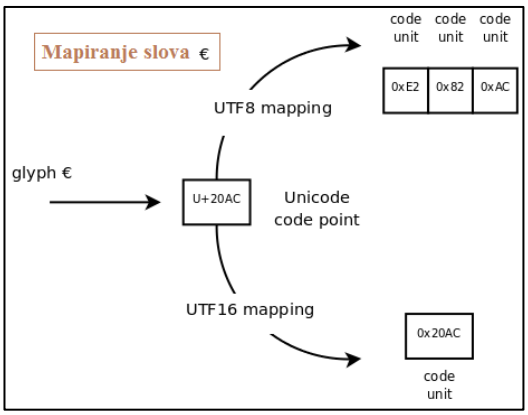
Prvenstveno da se izbjegnu greške, ali i da se optimizira veličina dokumenta razvijene su posebne transformacione šeme. One su omogućile da se koriste promjenljiva dužina karaktera. Tako se došlo do **unikod transformacionih formata UTF** (Unicode Transformation Format). To je način zapisa kodnih tačaka u standardu Unicode pomoću nizova 8-bitnih bajtova. Kodovanje zavisi od pozicije znaka i vrijednosti kodne tačke.

Transformaciona šema **UTF-8**, kao **dio unikod familije** radi tako da se karakter zapisuje **u jednom, dva ili tri bajta, u zavisnosti od toga o kom je karakteru riječ**. *Ova šema zadovoljava većinu potreba i na našim prostorima.*

Ovakav način zapisa i kodovanja omogućio je da se riješi problem zapisa skoro svih postojećih pisama[3].

Character	UTF-16	UTF-8	UCS-2
A	0041	41	0041
c	0063	63	0063
Ö	00F6	C3 B6	00F6
亜	4E9C	E4 BA 9C	4E9C
𝄞	D834 DD1E	F0 9D 84 9E	N/A

Ilustracija 8 Kodovanja znaka UCS-om i UTF transformacijama



Ilustracija 9 UTF mapiranje

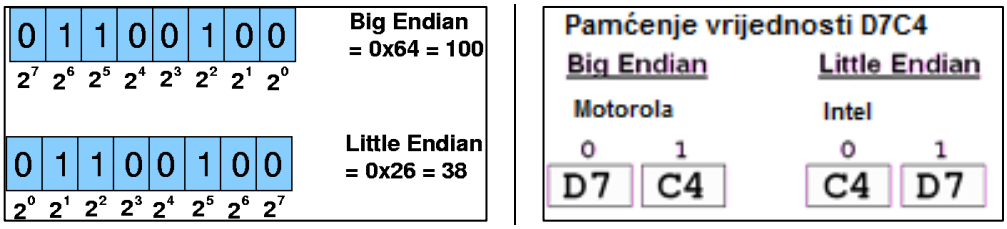
[3] postoje problemi kod nekih verzija kineskog (han) zapisa

Ako je znak u ASCII setu od 0 do 128 predstavlja se kao jedan bajt.
Svi drugi kodovi prikazani su tako da se jedan znak predstavlja kao niz od 2 do 4 bajta od kojih je svaki u rasponu od 128 do 255. U takvom nizu od dva ili više bajtova, svaki bajt ima MSB: najznačajniji bit postavljen na 1. Upravo ova 1 eliminiše probleme sa kontrolnim signalima i eliminacijom nosećih nula.

Bits	Last code point	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
7	U+007F	0xxxxxxx					
11	U+07FF	110xxxxx	10xxxxxx				
16	U+FFFF	1110xxxx	10xxxxxx	10xxxxxx			
21	U+1FFFFF	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx		
26	U+3FFFFFF	111110xx	10xxxxxx	10xxxxxx	10xxxxxx	10xxxxxx	
31	U+7FFFFFFF	1111110x	10xxxxxx	10xxxxxx	10xxxxxx	10xxxxxx	10xxxxxx

Ilustracija 10 Kreiranje kodova promjenljive dužine korišćenjem transformacionih formata

Za razjašnjenje bi trebali ući u konkretne detalje programiranja i protokola komunikacije (npr. definisati razloge zašto se premještaju „lijeve“ i „ desne“ težinske vrijednosti Big i Little Endian-a), a to ostavljamo kao programerski zadatak (npr. iz c-a) uz ilustraciju na sljedećoj slici koja to pokazuje:



Ilustracija 11 Različito definisanje najvažnijeg bita

UTF-8 je pogodan za korišćenje u izvornom (source) kodu programa ili u markup jezicima (HTML, XML, \LaTeX, ...), jer su standardne komande tih programskih/markup jezika uvijek ASCII, a tekst koji se koristi može da bude i ASCII i UTF-8. Na taj način se ne ometa rad programskog kompajlera ili parsera markup jezika, a omogućava se korišćenje višejezičke podrške.

Izvori i reference

https://razno.sveznadar.info/ [1, 2023]

IEEE Standard for Radix-Independent Floating-Point Arithmetic, The Institute of Electrical and Electronics Engineers, Inc, IEEE Standards Board, Approved September 10, 1987, Reaffirmed August 23, 1994

IEEE Standards Dictionary Online https://www.ieee.org/publications_standards/publications/subscriptions/prod/standards_dictionary.html [1, 2017]

Informatička abeceda: http://www.informatika.buzdo.com/index.html , [2,2016]

Mala kompjuteska enciklopedija

MKE

Banja Luka, 2023

Ovaj priručnik ima isključivo edukativnu namjenu.