Увод у архитектуру и организацију рачунара 1

Јован Самарџић, 13/2019

Професор: Стефан Мишковић

- дефиниције
- ознаке
- теореме
- докази
- примери

Година курса: 2020/21

Молим да ми све грешке пријавите преко мејла или друштвених мрежа.

1. Бројевни системи.

1.1 Бројевне системе делимо у две групе:

1) **непозиционе**: свака цифра, без обзира на место у запису, има исту вредност.
   ↳ нпр. римски бројни систем

2) **позиционе**: место цифре у запису утиче на вредност броја.
   ↳ нпр. декадни бројни систем
   ↳ могу имати: * фиксну основу (декадни, бинарни...) - њима се бавимо
   * променљиву основу (сат)

1.2 $(x)_n = \underbrace{x_{k-1} .. x_1 x_0}$ - број x је записан у основи n, помоћу k цифара
↳ дужина броја

x_i - цифра на i-тој позицији
x_{k-1} - цифра највеће тежине, x_0 - цифра најмање тежине

пр. 1: $(3125)_{10}$

Да би се дефинисао бројни систем са фиксном основом, довољно је навести: * вредност основе
* вредности цифара

пр. 2:
1) **декадни**: основа $n=10$, цифре $\{0,1,2,3,4,5,6,7,8,9\}$
2) **бинарни**: основа $n=2$, цифре $\{0,1\}$
3) **октални**: основа $n=8$, цифре $\{0,1,2,3,4,5,6,7\}$
4) **тробни**: основа $n=3$, цифре $\{0,1,2\}$
5) **хексадекадни**: основа $n=16$, цифре $\{0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F\}$
6) **негабинарни**: основа $n=-2$, цифре $\{0,1\}$
7) **негадекадни**: основа $n=-10$, цифре $\{0,1,2,3,4,5,6,7,8,9\}$
8) **са реалном основом**: нпр. основа $n=0.5$, цифре $\{0,1,2,3,4,5,6,7,8,9\}$
9) **балансирани тробни, bt**: основа $n=3$, цифре $\{-1,0,1\}$

1.3 **Мешовит број** је онај који има цифара различитих од 0 са обе стране зареза.

↳ начини записивања:

1) **уобичајени облик:** $(x)_n = \underbrace{x_{k-1} \dots x_1 x_0}_{\text{цео део}} . \underbrace{x_{-1} x_{-2} \dots x_{-l}}_{\text{разломљен део}}$

2) **фиксни зарез:** - $m.l$ значи укупно m цифара, од чега су l ($\leq m$) за децимале
- ако има више од m или више од $m-l$ „недецималних" цифара: не може
- ако има више од l децимала: одсечемо
- ако има више од l децимала: додамо нуле

пр. 3 Записујемо 135.43914 у фиксном зарезу:

8.5	135.43914
9.6	135.439140
8.6	не може
6.3	135.439
8.3	135.439

3) **покретни зарез, (f,e):** - запис $f\ n^e$, где је f-фракција, e-експонент, n-основа
- постоји ∞ записа за неки број, али издваја се **нормализован облик**

пр. 4 Декадни број 123.456 можемо записати као:

$(12.3456, 1)$
$(1.23456, 2)$ - нормализован
$(123456, -3)$
$\vdots$

1.4 **Превођење у декадни:** $(x_{k-1} \dots x_1 x_0 . x_{-1} \dots x_{-l})_n = x_{k-1} \cdot n^{k-1} + \dots + x_1 \cdot n + x_0 + x_{-1} \cdot n^{-1} + \dots + x_{-l} \cdot n^{-l} = \sum_{i=-l}^{k-1} x_i n^i$

пр. 5 $(233.12)_8 = 2 \cdot 8^2 + 3 \cdot 8 + 3 \cdot 1 \cdot 8^{-1} + 2 \cdot 8^{-2} = 128 + 24 + 3 + 0.125 + 0.03125 = (155.15625)_{10}$

1.5 **Превођење из декадног:** одвојено преведемо цео, па разломљени део и онда их спојимо.

* Желимо да преведемо **цео** број x из основе 10 у основу n.
Нека је $(x)_n = x_{k-1} \dots x_1 x_0$. $(x)_{10} = (x)_n \Rightarrow (x)_{10} = x_{k-1} \cdot n^{k-1} + \dots + x_1 \cdot n + x_0 \overset{/:n}{\Rightarrow} \frac{(x)_{10}}{n} = \underbrace{x_{k-1} \cdot n^{k-2} + \dots + x_1 \cdot n^0}_{(x')_{10}} + x_0 \cdot n^{-1}$

Дакле: $\frac{(x)_{10}}{n} = \underbrace{(x')_{10}}_{\text{цео}} + \underbrace{\frac{x_0}{n}}_{\text{разломљен}}$.

Алгоритам: Цео број $(x)_{10}$ поделимо са n. Остатак је цифра x_0, а на количник применимо исто.

* Желимо да преведемо **разломљен** број x из основе 10 у основу n.
Нека је $(x)_n = 0.x_{-1} \dots x_{-l}$. $(x)_{10} = (x)_n \Rightarrow (x)_{10} = x_{-1} \cdot n^{-1} + x_{-2} \cdot n^{-2} + \dots + x_{-l} \cdot n^{-l} \overset{/:n}{\Rightarrow} n \cdot (x)_{10} = x_{-1} \cdot n^0 + \underbrace{x_{-2} \cdot n^{-1} + \dots + x_{-l} \cdot n^{-(l-1)}}_{(x')_{10}}$

Дакле: $n \cdot (x)_{10} = \underbrace{x_{-1}}_{\text{цео}} + \underbrace{(x')_{10}}_{\text{разломљен}}$

Алгоритам: Разломљен број $(x)_{10}$ помножимо са n. Целобројни део разломка је цифра која иде на почетак, а на децимални део применимо исто.

↳ Напомена: може доћи до периодичног.

Пр. 6 1) $(3129)_{10} = (300321)_4$, јер:

$3129 = 782 \cdot 4 + 1$
$782 = 195 \cdot 4 + 2$
$195 = 48 \cdot 4 + 3$
$48 = 12 \cdot 4 + 0$
$12 = 3 \cdot 4 + 0$
$3 = 0 \cdot 4 + 3$

3129	782	195	48	12	3	0
1	2	3	0	0	3	

2) $(0.375) = (0.011)_2$, јер:

$0.375 \cdot 2 = 0.75 = 0.75 + 0$
$0.75 \cdot 2 = 1.5 = 0.5 + 1$
$0.5 \cdot 2 = 1 = 0.0 + 1$

0.375	0.75	0.5	0	
0	0	1	1	

↑ овде увек 0

1.6 Превођење из основе n у m:

I начин: користећи међупревод у декадни систем

II начин: директно - исто као и пре само ставимо m уместо 10 и дељење је у основи n. (тј. $(n)_m$)

Пр. 7 $(2301.32)_4 = (453.513)_6$: $(6)_{10} = (12)_4$, па делимо бројем $(12)_4$

* цео део:

$2301 : 12 = 131$ (остатак 3)
$131 : 12 = 10$ (остатак 11)

1	11	21	31	101	111
2	[12]	22	32	[102]	112
3	13	23	33	103	113
10	20	[30]	100	110	[120]

(налицемо до највећег и довршимо дељење)

$2301 = 131 \cdot 12 + 3$
$131 = 10 \cdot 12 + 11$
$10 = 0 \cdot 12 + 10$

2301	131	10	0
3	11	10	

читамо у основи 6

па је $(2301)_4 = (453)_6$

* разломљени део:

$0.32 \cdot 12$
130
+ 32
1110 → 11.10

$0.1 \cdot 12 = 1.2$ $0.2 \cdot 12 = 3.0$

1	11	21
2	[12]	22
3	13	23
10	20	[30]

$0.32 \cdot 12 = 11.10 = 0.1 + 11$
$0.1 \cdot 12 = 1.2 = 0.2 + 1$
$0.2 \cdot 12 = 3.0 = 0.0 + 3$

па је $(0.32) = (0.513)_6$

1.6 Специјални случајеви:

1) $m = n^k$ $(n<m)$ - групишемо по k цифара
2) $n = m^k$ $(n>m)$ - запишемо сваку цифру помоћу k цифара (из основе m).

Пр. 8 1) $(101101.01)_2 = (0010|1101.0100)_2 = (2D.4)_{16}$

2) $(2D.4)_{16} = (0010|1101.0100)_2 = (101101.01)_2$

2. Запис означених бројева.

2.1 Број је **неозначен** ако се за запис броја знак не узима у обзир.

↳ у рачунару: записују се на стандардан начин
интервал: $[0, 2^k-1]$, где је k број цифара.
вредност броја добија се превођем у декадни.

Број је **означен** ако се за запис броја знак узима у обзир.

↳ у рачунару: знак морамо представити бројевима, па зато постоје 4 записа:
1) **знак и апсолутна вредност**;
2) **непотпуни комплемент**;
3) **потпуни комплемент**;
4) **вишак k**.

Сада ћемо да испитујемо ове записе и операције у њима (за рачун само случај бинарних).

2.2 **Знак и апсолутна вредност**: на знак допишемо апсолутну вредност.

бин. ↳ опсег: $[-2^{k-1}+1,\ 2^{k-1}-1]$

↳ + је 0
− је n-1 (n – основа)

Дакле, за $X = \pm X_{k-2} \dots X_1 X_0$ $\Rightarrow$

$0X_{k-2} \dots X_1X_0$	$X>0$
$(n-1)X_{k-2} \dots X_1X_0$	$X<0$
$00\dots0$	$X=+0$
$(n-1)0\dots0$	$X=-0$

(двострука 0)

пр. 1 $(345)_{10}$: 0345
$(-345)_{10}$: 9345

* **конверзија између различитих дужина**: додамо нуле после знака.

пр. 2 $(-345)_{10}$ са 6 цифара је 900345. Али $(-345)_{10}$ са 3 цифре није могуће.

* **промена знака**: заменимо цифру за знак $(0 \leftrightarrow n-1)$

* **сабирање**:
1° + и + : само саберемо апс. вр. и знак је +.
2° − и − : само саберемо апс. вр. и знак је −.
3° + и − : знак је од оног са већом апс. вр., а апс. вр. је једнака разлици апс. вр. сабирака.
Може се десити да збир не може записати истим бројем цифара као сабирци. (прекорачење)

пр. 3 $(0\,001101)_2^7 + (0\,010011)_2^7 = (010000)_2^7$
$(1\,001101)_2^7 + (1\,010011)_2^7 = (110000)_2^7$
$(1\,0011)_2^5 + (0\,1000)_2^5 = (0\,0101)_2^5$
0100 + 0111 не може са 4 цифре, већ мора 5.

$$\begin{array}{r} 001101 \\ +\,010011 \\ \hline 100000 \end{array} \qquad \begin{array}{r} \overset{\cdot}{1}\overset{\cdot}{0}\overset{\cdot}{0}0 \\ -\,0011 \\ \hline 0101 \end{array}$$

* **одузимање**: своди се на сабирање и промену знака $(x-y = x+(-y))$

2.3 Непотпуни комплемент: позитивни исто као пре, негативни тако што све цифре комплементирамо. (заменимо комплементом до највеће цифре основе)

бин. ↳ опсег: $[-2^{k-1}+1,\ 2^{k-1}-1]$

Дакле, за $X = \pm X_{k-2} \dots X_1 X_0$ $\Rightarrow$

$$\begin{cases} 0X_{k-2}\dots X_1X_0, & X>0 \\ (n-1)\overline{X_{k-2}}\dots\overline{X_1}\,\overline{X_0}, & X<0 \\ 00\dots0, & X=+0 \\ (n-1)(n-1)\dots(n-1), & X=-0 \end{cases} \quad (\overline{X_i} = (n-1) - X_i) \quad (\text{двострука } 0)$$

пр. 4 $(345)_{10}$: 0345
$(-345)_{10}$: 9654

Комплементациона константа = $n^k - 1$ (негативан = const - позитиван)

пр. 5 За декадне бројеве на 4 места, const = 9999

* конверзија између различитих дужина: додамо цифре за знак

пр. 6 $(-345)_{10}$ са 6 цифара је 999654. Али $(-345)_{10}$ са 3 цифре није могуће.

* промена знака: комплементирамо све цифре.

* сабирање: две фазе: I) саберемо бројеве на класичан начин (чак и цифру за знак)
II) (евентуални) пренос на највећој позицији се дода на резултат (пренос је 0 или 1)
Може доћи до прекорачења: збир два поз. је нег. или збир два нег. је поз.

пр. 7 $00110 + 00111 = 01101$. Пренос је 0.
$11001 + 01110 = 1|00111$. Пренос је 1, па то додајемо: $00111 + 1 = 01000$.

$0100 + 0111 = 1011$. Ово је прекорачење, а није било преноса.
$10110 + 11111 = 1|10101$, па $10101 + 1 = 10110$. Ово није прекорачење, а било је преноса.

* одузимање: своди се на сабирање и промену знака $(x - y = x + (-y))$

2.4 Потпуни комплемент: позитивни исто као пре,
негативни тако што све цифре комплементирамо, а онда додамо 1 на најмању цифру.
(ако има пренос на крају) занемаримо га (некад на децимале) (не обавезно цео део)

бин. ↳ опсег: $[-2^{k-1}, 2^{k-1}-1]$

Дакле, за $X = \pm x_{k-2} \dots x_1 x_0$ $\Rightarrow$
$$\begin{cases} 0x_{k-2} \dots x_1 x_0 & , \quad X > 0 \\ (n-1)\overline{x_{k-2}} \dots \overline{x_1}\,\overline{x_0} + 1 & , \quad X < 0 \\ 00\dots0 & , \quad X = 0 \end{cases}$$
(јединствено 0)

пр. 8 $(345)_{10}$: 0345
$(-345)_{10}$: 9655

Комплементациона константа = n^k

пр. 9 За декадне бројеве на 4 места, const = 10 000

* конверзија између различитих дужина: додамо цифре за знак

пр. 10 $(-345)_{10}$ са 6 цифара је 999655. Али $(-345)_{10}$ са 3 цифре није могуће.

* промена знака: две фазе: I) комплементирамо све цифре. (бин.)
II) додамо 1 на резултат

пр. 11 $(0345)^4_{10} \rightarrow (9654)^4_{10} \xrightarrow{+1} (9655)^4_{10}$

* сабирање: саберемо бројеве на класичан начин (чак и цифру за знак)
(евентуални) пренос на највећој позицији се игнорише
Може доћи до прекорачења: збир два поз. је нег. или збир два нег. је поз.

пр. 12 $00110 + 00111 = 01101$. Пренос је 0.
$11001 + 01110 = 1|00111$ Пренос је 1, али га игноришемо, па је резултат 00111.

$0100 + 0111 = 1011$. Ово је прекорачење, а није било преноса.
$10110 + 11111 = 1|10101$ Ово није прекорачење, а било је преноса.
↓ игноришемо

* одузимање: своди се на сабирање и промену знака $(x - y = x + (-y))$

2.5 **Вишак k:** на потпуни комплемент додамо k (у одговарајућем бројевном систему).

Дакле, за $X = \pm x_{k-2} \dots x_1 x_0 \Rightarrow \begin{cases} 0x_{k-2} \dots x_1 x_0 + k, & X > 0 \\ (n-1)\overline{x_{k-2}} \dots \overline{x_1}\,\overline{x_0} + k, & X < 0 \\ 00\dots0, & X = 0 \end{cases}$ (јединствено 0)

пр. 13 за k=4: $(345)_{10}$: 0349
$(-345)_{10}$: 9659

* **конверзија између различитих дужина:** додамо цифре за знак

пр. 14 $(-345)_{10}$ са 6 цифара је 999659 (k=4). Али $(-345)_{10}$ са 3 цифре није могуће.

* **промена знака:** две фазе: I) запишемо га у потпуном комплементу (бин.)
II) додамо k на резултат

пр. 15 $(0345)^4_{10} \xrightarrow{\text{п.к.}} (9655)^4_{10} \xrightarrow{+4} (9659)^4_{10}$

* **сабирање:** саберемо бројеве на класичан начин (чак и цифру за знак)
(евентуални) пренос на највећој позицији се игнорише
одузмемо k на крају $(x'+y' = (x+k)+(y+k) = x+y+2k$, па да би рез. био у вишкд. одузмемо $k)$
Може доћи до прекорачења: збир два поз. је нег. или збир два нег. је поз.

* **одузимање:** своди се на сабирање и промену знака $x-y = x+(-y)$
додамо k на крају $(x'-y' = (x+k)-(y+k) = x-y$, па да би рез. био у вишку, додамо $k)$.

Зашто ово постоји? Користи се, нпр., када желимо да најмањи број буде записан нулама.

нпр. Опсег је $[-2^{k-1}, 2^{k-1}-1]$. Узмемо вишак 2^{k-1} и опсег постаје $[0, 2^k-1]$.
Тада су сви бр. позитивни, па је поређење лакше.

3. Množenje i deljenje brojeva.

3.1 Množenje neoznačenih brojeva: isto kao na papiru kad radimo. (uz to ovde gledamo binarno)

pr. 1

$$
\begin{array}{r}
1110 \cdot 1001 \\
\hline
1110 \\
0000 \\
0000 \\
1110 \\
\hline
1101110
\end{array}
$$

Množenje jeste komutativno, ali u računaru ih posmatramo odvojeno, tj. kao **množenik** i **množilac**.

Primetimo šta smo radili: 1° cifra množioca je 1 $\Rightarrow$ prepišemo množenik
2° cifra množioca je 0 $\Rightarrow$ ispišemo nule

Nakon svakog koraka uvučemo.

Ako množenik i množilac imaju po k cifara, proizvod ih ima najviše 2k.

Na osnovu toga, konstruišemo hardverski algoritam za množenje dva neoznačena cela binarna br.

Postavka: Imamo dva neoznačena cela bin. br. od k cifara. Za proizvod je dovoljno 2k bitova.
Prvi činilac, množenik, upišimo u registar M.
Drugi činilac, množilac, upišimo u registar P.
Potrebni su nam još registar A dužine k i registar C dužine 1.

Algoritam: Na početku: u M je upisan množenik, a u P množilac. Označimo poslednji bit P sa P_0.
u A i C sve nule.

k puta: I) 1° $P_0 = 0 \Rightarrow$ ništa
2° $P_0 = 1 \Rightarrow A = A + M$, a (eventualni) prenos ide u C.

II) Šiftujemo udesno „spojeni" registar CAP. (logički)

Proizvod čitamo iz „spojenog" registra AP od 2k bitova.

pr. 2 Množimo neoznačene binarne br.

$(112)_{10} = (01110000)_2^8 \rightarrow$ M
$(9)_{10} = (00001001)_2^8 \rightarrow$ P

Rezultat: $(1111110000)_2 = (1008)_{10}$

Korak	C	A	P	Komentar
0	0	00000000	00001001	Vrši se inicijalizacija.
1	0	01110000	00001001	$P_0 = 1$, vrši se sabiranje $A = A + M$.
	0	00111000	00000100	Registar CAP se logički pomera udesno.
2	0	00111000	00000100	$P_0 = 0$, ne vrši se nikakva akcija.
	0	00011100	00000010	Registar CAP se logički pomera udesno.
3	0	00011100	00000010	$P_0 = 0$, ne vrši se nikakva akcija.
	0	00001110	00000001	Registar CAP se logički pomera udesno.
4	0	01111110	00000001	$P_0 = 1$, vrši se sabiranje $A = A + M$.
	0	00111111	00000000	Registar CAP se logički pomera udesno.
5	0	00111111	00000000	$P_0 = 0$, ne vrši se nikakva akcija.
	0	00011111	10000000	Registar CAP se logički pomera udesno.
6	0	00011111	10000000	$P_0 = 0$, ne vrši se nikakva akcija.
	0	00001111	11000000	Registar CAP se logički pomera udesno.
7	0	00001111	11000000	$P_0 = 0$, ne vrši se nikakva akcija.
	0	00000111	11100000	Registar CAP se logički pomera udesno.
8	0	00000111	11100000	$P_0 = 0$, ne vrši se nikakva akcija.
	0	00000011	11110000	Registar CAP se logički pomera udesno.

3.2 Множење означених бројева: користимо хардверски Бутов алгоритам - бројеви су записани у потп. компл.

(у потп. компл.)

Поставка: Имамo два означена цела бин. бр. од k цифара. За производ је довољно 2k битова.
Први чинилац, множеник, упишимо у регистар **M**.
Други чинилац, множилац, упишимо у регистар **P**.
Потребни су нам још регистар **A** дужине k и регистар P_{-1} дужине 1.

Алгоритам: На почетку: у M је уписан множеник, а у P множилац. Означимо последњи бит P са P_0.
у A и P_{-1} све нуле.

k пута:
I) 1° $P_0P_{-1} = 00$ или $P_0P_{-1} = 11$ $\Rightarrow$ ништа
2° $P_0P_{-1} = 01 \Rightarrow A = A + M$
3° $P_0P_{-1} = 10 \Rightarrow A = A - M$

II) Шифтујемо удесно „спојени" регистар APP_{-1}. (аритметички)

Производ читамо из „спојеног" регистра AP од 2k битова.

3.3 Модификован Бутов алгоритам: (n/2) за ефикасније множење означених бројева. Показујемо на примеру -28 и 111.

1) Ако множеник има 2k битова, онда множилац има k. Узмимо да дужине буду 16 и 8.

2) $(-28)_{10} = (1111111111100100)_2^{16}$
$(111)_{10} = (01101111)_2^{8}$ - ово је Бутов множилац.
идеја је да уклонимо низове јединица: -1 → почетак серије јединица
+1 → прва нула након серије јединица
0 → све остало

0	1	1	0	1	1	1	1
+1	0	-1	+1	0	0	0	-1

→ **Бутов кодирани множилац**

3) Издвајамо парове здесна налево и придружујемо број сваком пару: $(a_{2i+1}, a_{2i}) \rightarrow 2a_{2i+1} + a_{2i} \in \{-2,-1,0,1,2\}$

У овом примеру: $(a_1, a_0) = (0,-1) \rightarrow -1 = v_0$, $(a_3, a_2) = (0,0) \rightarrow 0 = v_1$, $(a_5, a_4) = (-1,+1) \rightarrow -1 = v_2$, $(a_7, a_6) = (+1,0) \rightarrow 2 = v_3$

4) За свако $i \in \{0,1,2,3\}$ прво померимо множеник за 2i бита улево и такав број помножимо вредношћу пара.
Важи:
1° $v_i = 2$: множење се своди на шифтовање множеника улево за једно место.
2° $v_i = 1$: множеник се не мења.
3° $v_i = 0$: тај резултат је 0.
4° $v_i = -1$: множење се своди на мењање знака множенику (инверт цифре и +1)
5° $v_i = -2$: множење се своди на мењање знака множенику, а онда га шифтујемо улево за 1.
(исти ефекат се постиже и обрнутом редоследу)

Све те производе саберемо и тиме добијамо коначан резултат.

У овом примеру:

i	v_i	померен множеник	међупроизвод
0	-1	11111111 11100100	00000000 00011100
1	0	11111111 10010000	00000000 00000000
2	-1	11111110 01000000	00000001 11000000
3	-2	11111001 00000000	11110010 00000000

$\Rightarrow$ Резултат је: $(11110011\ 11011100)_2^{16}$ што је $(-3108)_{10}$

3.4 **Дељење неозначених бројева:** исто као на папиру кад радимо. (уз то овде гледамо бинарно)

пр. 3 $110101 : 101 = 1010$. Дакле, количник је

```
110101 : 101 = 1010
-101
 110
-101
  11
```

Приметимо шта смо радили: 1° цифре делиоца се садрже у „тренутним" цифрама дељеника ⇒ пишемо 1.
2° цифре делиоца се не садрже у „тренутним" цифрама дељеника ⇒ пишемо 0.

Након сваког корака на тренутни остатак дописемо наредну цифру дељеника.

Поступак се понавља док не искористимо све цифре дељеника.

На основу тога, конструишемо **хардверски алгоритам** за дељење два неозначена цела бинарна бр.

Поставка: Имамо два неозначена цела бин. бр. од k цифара.
Дељеник упишимо у регистар P. Означимо последњи бит P са P_0.
Делилац упишимо у регистар M.
Потребан нам је још регистар A дужине k.

Алгоритам: На почетку: у P је уписан дељеник, а у M делилац.
у A све нуле.

k пута: I) Шифтујемо улево „спојени" регистар AP.

II) 1° A<M: ништа
2° A≥M: A = A-M и у P_0 уписујемо 1.

Количник читамо из регистра P, а остатак из регистра A.

пр. 4 Делимо неозначене бинарне бр.

$(24)_{10} = (00011000)_2^8$ → P
$(9)_{10} = (00001001)_2^8$ → M

Резултат: $(00000010)_2^8$

и остатак $(00000110)_2^8$.

Korak	A	P	Komentar
0	00000000	00011000	Vrši se inicijalizacija.
1	00000000	00110000	Registar AP se pomera ulevo.
	00000000	00110000	Kako je $A < M$, ne vrši se nikakva akcija.
2	00000000	01100000	Registar AP se pomera ulevo.
	00000000	01100000	Kako je $A < M$, ne vrši se nikakva akcija.
3	00000000	11000000	Registar AP se pomera ulevo.
	00000000	11000000	Kako je $A < M$, ne vrši se nikakva akcija.
4	00000001	10000000	Registar AP se pomera ulevo.
	00000001	10000000	Kako je $A < M$, ne vrši se nikakva akcija.
5	00000011	00000000	Registar AP se pomera ulevo.
	00000011	00000000	Kako je $A < M$, ne vrši se nikakva akcija.
6	00000110	00000000	Registar AP se pomera ulevo.
	00000110	00000000	Kako je $A < M$, ne vrši se nikakva akcija.
7	00001100	00000000	Registar AP se pomera ulevo.
	00000011	00000001	Kako $A \geq M$, važi $A = A - M$ i $P_0 = 1$.
8	00000110	00000010	Registar AP se pomera ulevo.
	00000110	00000010	Kako je $A < M$, ne vrši se nikakva akcija.

3.5 Дељење означених бројева: користимо хардверски алгоритам у ком су бројеви су записани у потп. компл.

Поставка: Имамо два означена цела бин. бр. (у потп. компл.) и то дељеник $2k$, а делилац k битова.
Дељеник упишимо у „спојени" регистар AP. (дакле ако је дељеник поз., A почиње нулама)
Означимо последњи бит P са P_0.
Делилац упишимо у регистар M.

Алгоритам: На почетку: у AP је уписан дељеник, а у M делилац.

k пута: I) Шифтујемо улево „спојени" регистар AP.

II) 1° A и M (упоредимо 1. бит) истог знака $\Rightarrow$ $A = A - M$, а у P_0 се уписује 1. (*)
2° A и M различитих знакова $\Rightarrow$ $A = A + M$, а у P_0 се уписује 1. (*)

(*) Ако:
а) након сабирања/одузимања, A мења знак
б) Ако након последњег корака (након сабирања/одузимања), $A \neq 0$
онда ипак вратимо на претх. стање (у A упишемо старо A, а у P_0 упишемо 0).

Количник читамо из регистра P, а остатак из регистра A.
Пазимо на знак.

пр. 5 Делимо означене бинарне бр.

$(24)_{10} = (00000000\ 00011000)_2^{16} \rightarrow$ AP
$(-9)_{10} = (11110111)_2^8 \rightarrow$ M

Резултат: $(00000010)_2^8$

и остатак $(00000110)_2^8$.

Korak	A	P	Komentar
0	00000000	00011000	Vrši se inicijalizacija.
1	00000000 00000000	00110000 00110000	Registar AP se pomera ulevo. Ne vrši se nikakva akcija.
2	00000000 00000000	01100000 01100000	Registar AP se pomera ulevo. Ne vrši se nikakva akcija.
3	00000000 00000000	11000000 11000000	Registar AP se pomera ulevo. Ne vrši se nikakva akcija.
4	00000001 00000001	10000000 10000000	Registar AP se pomera ulevo. Ne vrši se nikakva akcija.
5	00000011 00000011	00000000 00000000	Registar AP se pomera ulevo. Ne vrši se nikakva akcija.
6	00000110 00000110	00000000 00000000	Registar AP se pomera ulevo. Ne vrši se nikakva akcija.
7	00001100 00000011	00000000 00000001	Registar AP se pomera ulevo. Vrši se oduzimanje $A = A - M$ i $P_0 = 1$.
8	00000110 00000110	00000010 00000010	Registar AP se pomera ulevo. Ne vrši se nikakva akcija.

4. Бинарно кодирање.

4.1 **BCD код** је, у општем случају, било која ф-ја која сваку декадну цифру слика у низ бинарних цифара.
↳ уводимо их да бисмо могли да запишемо велике бројеве

Иако је дозвољено да ти низови буду разних дужина, ми ћемо гледати **кодове где су те дужине const** (≥ 4).
Такође, дозвољено је да се више декадних цифара слика у исти низ, али ми ћемо гледати **1-1 кодове**.
Поред ова два, пожељно је да код поседује нека од наредних својстава:

1) **парност** - парним цифрама одговарају парни кодови, а непарним непарни.
2) **комплементарност** - одг. битови кодова a и $\bar{a}$ су комплементарни (тј. збир кодова компл. цифри је 11...11).
3) **тежински** - ако је $X \mapsto X_3X_2X_1X_0$, постоје c_3, c_2, c_1, c_0 (**тежине**) тд. $X = c_0x_0 + c_1x_1 + c_2x_2 + c_3x_3$.
4) **највећој цифри придружујемо „највећи" код** - тј. кад бисмо тај код гледали као број, имао би највећу вредност.

пр. 1 Неки BCD кодови су: (сви ови су дужине 4)

1) **8421** - декадна цифра се преведе у бин. бр. и допунимо водећим нулама до дужине.
↳ 1), 3), 4), али не 2)
2) **вишак 3** - на декадну цифру додамо 3, па онда урадимо исто као пре.
↳ 2), 4), али не и 1), 3)
3) **циклички код** - кодови суседних дек. циф. разликују за тачно 1 бит. (има их више)
↳ овај из табеле не задовољава ништа.

цифра	8421	вишак 3	циклички
0	0000	0011	0001
1	0001	0100	0101
2	0010	0101	0111
3	0011	0110	1111
4	0100	0111	1110
5	0101	1000	1100
6	0110	1001	1000
7	0111	1010	1001
8	1000	1011	1011
9	1001	1100	0011

4.2 **Грејов код** је било која 1-1 ф-ја која сваки цео бр. из $[0, 2^k-1]$ слика у бинарну ниску,
али тако да се вредности ф-ја суседних бројева разликују за тачно 1 бит.
Очигледно, Грејов код није јединствен.

пр. 2 $g(n) = (n)_2^k \oplus \left(\left\lfloor \frac{n}{2} \right\rfloor\right)_2^k$, где је $\oplus$ **ексклузивна дисјункција**, а $\lfloor\ \rfloor$ је **цео део**.

↳ нпр. $g(2) = (2)_2^4 \oplus \left(\left\lfloor \frac{2}{2} \right\rfloor\right)_2^4 = (2)_2^4 \oplus (1)_2^4 = 0010 \oplus 0001 = 0011$
$g(3) = (3)_2^4 \oplus \left(\left\lfloor \frac{3}{2} \right\rfloor\right)_2^4 = (3)_2^4 \oplus (1)_2^4 = 0011 \oplus 0001 = 0010$ (заиста се разликују за тачно 1 бит)

Овај код се може и лепше записати: $g_i = \begin{cases} a_i, & i = k-1 \\ a_i \oplus a_{i+1}, & i < k-1 \end{cases}$; $a_i = \begin{cases} g_i, & i = k-1 \\ g_i \oplus a_{i+1}, & i < k-1 \end{cases}$

нпр. $a = 1001$: $g_3 = a_3 = 1$, $g_2 = a_2 \oplus a_3 = 1$, $g_1 = a_1 \oplus a_2 = 0$, $g_0 = a_0 \oplus a_1 = 1$

, тј. $a_3 = g_3 = 1$, $a_2 = g_2 \oplus a_3 = 0$, $a_1 = g_1 \oplus a_2 = 0$, $a_0 = g_0 \oplus a_1 = 1$

4.3 Декадни бројеви у BCD коду могу бити записани у:

Леви полубајт - говори да је реч о цифри
Десни полубајт - вредност те цифре

1) **непакованом запису** - свака цифра у посебан бајт.
   - 1) EBCDIC код - у леви полубајт пишемо F = 1111
   - 2) ASCII - у леви полубајт пишемо 3 = 0011

   За оба важи да ако је број означен, у бајту последње цифре пишемо C за поз., а D за нег.

   пр.3 неозначен декадни број 1579: EBCDIC = F1 F5 F7 F9 (пар = бајт)
   ASCII = 31 35 37 39
   означен декадни број 1579: EBCDIC = F1 F5 F7 C9
   означен декадни број -1579: ASCII = 31 35 37 D9

2) **пакованом запису** - по две цифре у једном бајту. (по једна у сваком полубајту)
   Ако је број означен, у последњем полубајту пишемо C за поз., а D за нег.
   Ако има непаран бр. цифара, додамо 0.
   EBCDIC и ASCII се овде поклапају. (пошто не морамо да истичемо да је у питању цифра)

   пр.4 неозначен декадни број 1579: EBCDIC = 15 79
   ASCII = 15 79
   означен декадни број 1579: 01 57 9C (додали 0)
   означен декадни број -12345: 12 34 5D

Промена знака: у оба записа се изводи тако што напишемо C уместо D, тј. обрнуто.

4.4 **Чен-Хо код** кодира тројку декадних цифара са 10 битова.
Постоји јер штеди меморију у односу на BCD (нпр. за 3 декадне цифре, Чен-Хо користи 10, а BCD 4·3=12 битова).

Постоје табела за кодирање (aei) и табела за декодирање (pqrtu) између 8421 и Чен-Хо.

пр.5 1) 918 је у 8241: 1001 0001 1000 = abcdefghijkl ⇒ aei = 101

Читамо табелу за кодирање (ред где је 101): pqrstuvwxy = 111d01hfgl = 1111011000

2) Дат је низ: pqrstuvwxy = 1111011000 ⇒ pqrtu = 11101

Читамо табелу за декодирање (ред где је 11101): abcdefghijkl = 100s0wxv100g = 1001 0001 1000 = 918

Ако је бр. цифара 3k, радимо k пута (за сваку тројку цифара посебно).
Ако бр. цифара није дељив са 3, додамо нуле на крај.

4.5 **DPD кодирање** је исто као Чен-Хо, само има друге табеле. (које су нешто ефикасније)

4.6 **Аритметика у BCD коду:** за означене бројеве сводимо на знак и апс. вр. Зато испитујемо неозначене.

Уопштено: I) сабирамо/одузимамо цифре (као неозначене бр.) на одг. позицијама и бележимо преносе. (ова фаза је иста за све)

II) вршимо одређене корекције (које зависе од кода у ком рачунамо).
То радимо ако, нпр. при збиру, добијемо четворку која је ≥ 10, тј. није цифра.

* **у запису 8421:**

  * **Сабирање:** показујемо на примеру.

I) Сабирамо: $(6841)_{10} \overset{8421}{=} 0110\ 1000\ 0100\ 0001$ и $(2893)_{10} \overset{8421}{=} 0010\ 1000\ 1001\ 0011$ као неозначене бр.
При томе, бележимо преносе „између четворки."

A		0110	1000	0100	0001
B	+	0010	1000	1001	0011
P'	0*	1	0	0	0
C'		1001	0000	1101	0100

$P' = \{p_i' \in \{0,1\} \mid p_i'$ - пренос са i-1 четворке на i-ту, $p_0' = 0\}$
$C' = A + B$
* ако би било $p_n' = 1 \Rightarrow$ прекорачење

II) Означимо најмању четворку са c_0'.
1° $c_0' \geq (10)_{10} = (1010)_2$ или $p_1' = 1$, корекција је $k_0 = (0110)_2$.
2° иначе, корекција је $k_0 = (0000)_2$.
Саберемо $c_0' + k_0 + \underset{0}{p_0''}$ и тај пренос означимо $p_1'' \in \{0,1\}$. Подразумевамо $p_0'' = 0$.

Правило за корекције:
1° $c_i' + p_i'' \geq (1010)_2$
2° $p_{i+1}' = 1$
$\Rightarrow k_i = (0110)_2 = (6)_2$ (пренос 1 је 16 а $16-10=6$)
иначе: $k_i = (0000)_2$

$c_i = c_i' + p_i'' + k_i$

Дакле:

P'	0	1	0	0	0
C'		1001	0000	1101	0100
P''	0	0	1	0	0
K		0000	0110	0110	0000
C		1001	0111	0011	0100

0: $p_0'' = 0$
$c_0' + p_0'' < (1010)_2 \wedge p_1' = 0 \Rightarrow k_0 = (0000)_2 \Rightarrow p_1'' = 0$
1: $c_1' + p_1'' \geq (1010)_2 \Rightarrow k_1 = (0110)_2 \Rightarrow p_2'' = 1$
2: $c_2' + p_2'' < (1010)_2$, али $p_3' = 1 \Rightarrow k_2 = (0110)_2 \Rightarrow p_3'' = 0$
3: $c_3' + p_3'' < (1010)_2 \wedge p_3' = 0 \Rightarrow k_3 = (0000)_2 \Rightarrow p_3'' = 0$

  * **Одузимање:** може на два начина - слично као пре или свођењем на сабирање

1. начин: I) исто, само сад одузимамо (као неозначене бр.).
При томе, бележимо позајмице „између четворки."

II) **Правило за корекције:**
1° $p_{i+1}' = 1 \Rightarrow k_i = (0110)_2$
2° $p_{i+1}' = 0 \Rightarrow k_i = (0000)_2$
(овде нема оних p'') ($p_0' = 0$)

$c_i = c_i' - k_i$

2. начин: 1. и умањеник и умањилац запишемо у потпуном комплементу. (умањеник - поз. умањилац - нег.)
2. применимо алгоритам за сабирање у 8421. (игноришу се преноси код прекорачења)
3. обришемо водеће нуле. (тиме пребацимо из п.к. у декадни)

↑ мора бити наглашено да су бројеви записани бар једном цифром више него у декадном (због цифре за знак)

* У запису вишак 3:

* Сабирање:

I) идентично као у запису 8421. (само су бројеви записани у вишку 3)

II) Правило за корекције: $1^\circ\ p'_{i+1} = 1 \Rightarrow k_i = (0011)_2\ (=(3)_{10})$
$2^\circ\ p'_{i+1} = 0 \Rightarrow k_i = (1101)_2\ (=(13)_{10})$
(овде нема оних p'') $(p'_0 = 0)$

Ако би било $p'_n = 1 \Rightarrow$ прекорачење
↑ пре корекција

* Одузимање:

мора бити наглашено да су бројеви записани бар једном цифром више него у декадном (због цифре за знак)

1. и умањеник и умањилац запишемо у потпуном комплементу.
2. применимо алгоритам за сабирање у вишку 3.
3. обришемо водеће нуле. (тиме пребацимо из п.к. у декадни)

5. Запис реалних бројева.

5.1 Реални бројеви се могу записати на два начина:

1) у непокретном зарезу: непрактично (због записа и због аритметике).

2) у покретном зарезу: практичнији / флексибилнији.
сваки број запишемо као: $\pm f \cdot n^e$, где је f-фракција, e-експонент
$f_0.f_1 \dots f_p$ $(f_0 \neq 0)$ - нормализован облик

Укупан број цифара фракције је прецизност, p.
Некада долази до одсецања/заокруживања (ако је p мало)

5.2 Запис са бинарном основом на старијим рачунарима: увек бинарно

1) једнострука тачност:

1 + 8 + 23 = 32 bit

* знак: позитиван = 0, негативан = 1.
* експонент: на 8 битова и то у запису вишак 128.
* фракција: $0.f_0f_1 \dots f_{p-1}$, где $f_0 \neq 0 \Rightarrow f_0 = 1$. То значи да увек почиње са 0.1, па пишемо само $f_1 \dots f_{p-1}$.

2) двострука тачност:

1 + 8 + 55 = 64 bit

* све исто, само још 32 додатна бита за фракцију.

пр. 1 $(-15)_{10} = (-1111)_2 = (-0.1111)_2 \cdot 2^4$

знак: 1
експонент: $4 + 128 = (132)_{10} = (10000100)_2$
фракција: 1110...0

$\Rightarrow$ 1 10000100 1110...0

5.3 Типови заокруживања:
1) заокруживање на парну цифру: 1.212 → 1.21 ; 1.218 → 1.22 ; 1.225 → 1.22 ; 1.235 → 1.24
2) заокруживање ка $+\infty$: 1.21 → 1.3 ; 1.29 → 1.3 ; -1.21 → -1.2 ; -1.29 → -1.2
3) заокруживање ка $-\infty$: 1.21 → 1.2 ; 1.29 → 1.2 ; -1.21 → -1.3 ; -1.29 → -1.3
4) заокруживање ка 0: 1.21 → 1.2 ; 1.29 → 1.2 ; -1.21 → -1.2 ; -1.29 → -1.2

Типови грешака: Заокружимо 0.0574367 на $5.47 \cdot 10^{-2}$. Разлика је 0.0000367
1) ULP грешка = 0.367
2) Релативна грешка = $|0.0000367| / |0.0574367| = 0.0006$

Цифре чувари - нпр. Одузимамо заокружене вредности од 10.1 и 9.93
$1.01 \cdot 10^1 - 0.99 \cdot 10^1 = 0.02 \cdot 10^1$, али $1.010 \cdot 10^1 - 0.993 \cdot 10^1 = 0.017 \cdot 10^1$

Тачно заокружене операције: када бисмо прво израчунали разлику, па тек онда заокружили.

5.4 **IEEE 754** је стандард за запис реалних бројева.
Подржава запис бројева са декадном и бинарном основом, а могући су записи 1, 2, 4 - струке тачности.

* **IEEE 754 са декадном основом** - основа (експонента) увек декадна,
фракција декадна (DPD кодирање) или бинарна (BID кодирање).
описаћемо оба типа и то у једнострукој тачности.

* **DPD кодирање:** и основа и фракција су декадне.

Пр. 2 Желимо да кодирамо број 123.4 по IEEE 754 са декадном основом при DPD кодирању.

1. Запишемо број тако. је фракција целобројна: $1234 \cdot 10^{-1}$
Затим фракцију запишемо на 7 цифара: 0001234 (ако већ има више од 7 ⇒ заокруживање)
И онда је раставимо на 1+3+3: 0 (3.) 001 (2.) 234 (2.)
2. Обе тројке кодирамо преко DPD табеле: 001 → 0000000001, 234 → 0100110100
3. Прву цифру (0) кодирамо тако што: 1° [0,7] - допунимо до тројке (2→010, 5→101): 0 → 000
2° 8→0 (само једна цифра)
3° 9→1 (само једна цифра)
4. Експонент (-1) записујемо бинарно на 8 места са увећањем 101: $-1+101 = 100 = (01100100)_2$
5. Правимо комбинацију: 1° 2 из експ. + тробитни код + преосталих 6 експ.: 01 000 100100
2°,3° 11 + 2 из експ. + једнобитни код + преосталих 6 експ.
6. Знак: позитиван = 0, негативан = 1: $123.4 > 0 \Rightarrow 0$

Коначан запис: 0 (6° знак) 01000100100 (5° комбинација) 0000000001 0100110100 (2° остатак фракције у DPD) (1+11+20 = 32)

* **BID кодирање:** основа декадна, фракција бинарна.

Пр. 3 Желимо да кодирамо број -14.37 по IEEE 754 са бинарном основом при BID кодирању.

1. Запишемо број тако. је фракција целобројна: $-1437 \cdot 10^{-2}$
Затим фракцију преведемо у бинарни систем: $-1437 \cdot 10^{-2} = (-10110011101)_2 \cdot 10^{-2}$
Затим фракцију запишемо на 23 цифре*: 00000000000010110011101
2. Експонент (-2) записујемо бинарно на 8 места са увећањем 101: $-2+101 = 99 = (01100011)_2$
3. Правимо комбинацију: експонент + 3 из фракције
4. Знак: позитиван = 0, негативан = 1: $-14.37 < 0 \Rightarrow 1$

Коначан запис: 1 (4° знак) 01100011000 (3° комбинација) 00000000010110011101 (1° остатак фракције у BID) (1+11+20 = 32)

* - погледати у скрипти за 24 цифре!!!

*Специјалне вредности: када се врши декодирање (супротно од ових алгоритама), морамо прво да се уверимо да није записана ни једна од наредних вредности, па тек онда декодирамо.
(исте и за DPD и BID)

1. Нула: коначна вредност или нека баш мала вр. која се не може записати (поткорачење).
постоје: позитивна нула, +0: фракција = 0 , 0 XX$\overline{000}$XXXXXX 00...00, 0 XXXXXXX$\overline{000}$ 00...00
негативна нула, -0: фракција = 0 , 1 XX$\overline{000}$XXXXXX 00...00, 1 XXXXXXX$\overline{000}$ 00...00
DPD BID

2. Бесконачност: за превелике бројеве (у обе крајности) и резултате неких аритм. операција (5:0 = +∞)
постоје: позитивна бесконачност, +∞: комб. почиње са 11110, 0 11110X...X X...X
негативна бесконачност, -∞: комб. почиње са 11110, 1 11110X...X X...X

3. NaN вредности: постоје: сигнални, sNaN: комб. почиње са 111111, X 111111XXXXX X...X
↳ грешке при иницијализацији или конверзији.
тихи, qNaN: комб. почиње са 111110, X 111110XXXXX X...X
↳ грешке при аритметичким операцијама.

5.5 *IEEE 754 са бинарном основом

- основа (експонента) увек бинарна,
фракција увек бинарна
запис у три дела: знак, експонент, фракција.
описаћемо и једноструку и двоструку тачност.

*Специјалне вредности:

1. Нула: све нуле и у експоненту и у фракцији, а знак 0 или 1.
2. Бесконачност: све јединице у експоненту и све нуле у фракцији, а знак 0 или 1.
3. NaN вредности: све јединице у експоненту, али: sNaN: први бит фрак. је 0, а у остатку бар један ≠0.
qNaN: први бит фрак. је 1, а остали произвољни.

*Једнострука тачност:

пр. 4 Желимо да кодирамо број 13.25 по IEEE 754 са бинарном основом при 1-струкој тачности.

1. Декадни број преведемо у бинарни: $(13.25)_{10} = (1101.01)_2$
Затим га доведемо у облик 1.f: $(1101.01)_2 = (1.10101)_2 \cdot 2^3$
Издвајамо децимални део фракције: 10101 (део 1. се увек појављује па га изостављамо из записа)

2. Експонент (3) записујемо бинарно на 8 места са увећањем 127: $3+127 = 130 = (10000010)_2$

3. Знак: позитиван = 0, негативан = 1: $13.25 > 0 \Rightarrow 0$

Коначан запис: 0 10000010 10101000000000000000000 (1+8+23=32)
знак експонент фракција и остатак су нуле

Субнормални бројеви: они који су веома блиски нули.
све нуле у експоненту. Када се декодира: фракција је облика 0.f
експонент је -126.

* Двострука тачност:

пр. 5 Желимо да кодирамо број 13.25 по IEEE 754 са бинарном основом при 2-струкој тачности.

1. Декадни број преведемо у бинарни: $(13.25)_{10} = (1101.01)_2$
   Затим га доведемо у облик 1.f: $(1101.01)_2 = (1.101101)_2 \cdot 2^3$
   Издвајамо децимални део фракције: 10101 (део 1. се увек појављује па га изостављамо из записа)

2. Експонент (3) записујемо бинарно на 11 места са увећањем 1023: $3+1023 = 1026 = (10000000010)_2$

3. Знак: позитиван = 0, негативан = 1: $13.25 > 0 \Rightarrow 0$

Коначан запис: 0 10000000010 10101000...000 $(1+11+52=64)$
(знак, експонент, фракција и остатак су нуле)

Субнормални бројеви: све нуле у експоненту. Када се декодира: фракција је облика 0.f
експонент је -1022.

5.6 Описујемо алгоритме за аритметичке операције у IEEE 754 са бинарном основом у 1-струкој тачности.
Пре било које операције морамо се уверити да нема спец. вредности.
После операције, може доћи до прекорачења или поткорачења - тада рез. „прогласимо" за ∞, тј 0.

*Сабирање: Прво доведемо бројеве на исти експонент и то онај већи.

$$f_1 \cdot 2^{e_1} + f_2 \cdot 2^{e_2} = f_1 \cdot 2^{e_1} + f_2' \cdot 2^{e_1} = (f_1 + f_2) \cdot 2^{e_1}$$

пр. 6 Сабирамо: a_1= 0 10000010 10011 0...0, a_2= 0 10000001 10110 0...0
Оба су поз. па је збир поз., па му је први бит 0.
a_1 има већи експонент, па је експонент збира 10000010.
Морамо да прилагодимо фракцију од a_2. Експ. a_2 је за 1 мањи од експ. $a_1 \Rightarrow f_{a_2}' = 0.11011$
($f_{a_2} = 1.1011$) (f_{a_1} се не мења $\Rightarrow f_{a_1} = 1.10011$)
$f = f_{a_1} + f_{a_2}' = 10.0111$. То морамо да нормализујемо: $f = (1.00111)_2 \cdot 2^1$
Морамо и експонент да повећамо за 1 $\Rightarrow e = 10000011$.

Дакле запис збира је: 0 10000011 00111 0...0

*Одузимање: Прво доведемо бројеве на исти експонент и то онај од умањеника.

$$f_1 \cdot 2^{e_1} - f_2 \cdot 2^{e_2} = f_1 \cdot 2^{e_1} - f_2' \cdot 2^{e_1} = (f_1 - f_2') \cdot 2^{e_1}$$

(Може и свођењем на сабирање)

*Множење: $(f_1 \cdot 2^{e_1}) \cdot (f_2 \cdot 2^{e_2}) = (f_1 f_2) \cdot 2^{e_1+e_2}$ (водити рачуна да су e_1, e_2 у вишку 127!)

*Дељење: $(f_1 \cdot 2^{e_1}) / (f_2 \cdot 2^{e_2}) = (f_1 / f_2) \cdot 2^{e_1-e_2}$ (водити рачуна да су e_1, e_2 у вишку 127!)

6. Запис бројева помоћу остатака.

Увод: a, b су **конгруентни по модулу** m, $a \equiv b \pmod m$, ако дају исти остатак при дељењу са m.

$RBS(m_n|\dots|m_2|m_1)$ означава **бројчани систем са остацима** $m_n, \dots, m_2, m_1$.

пр. 1 $(62)_{10} = (8|6|6)_{RBS(9|8|7)}$, јер су то редом остаци при дељењу 62 са 9, 8 и 7.

Како би се постигла јединственост записа, уводимо два додатна својства: 1. $NZD(m_i, m_j)=1$ $(i \neq j)$
2. $m_n > \dots > m_2 > m_1$

Овако можемо представити било којих $m = m_n \cdot \dots \cdot m_1$ узастопних бројева.
Од посебног значаја су: 1) $[0, m-1]$ - користимо за неозначене.
2) $[-\frac{m}{2}, \frac{m}{2}-1]$ - користимо за означене.

6.1 Модуларна аритметика:

*запис негативних бројева: пр. 2 Желимо да запишемо $(-538)_{10}$ у $RBS(9|7|4)$.

1. Урадимо за позитиван случај: $(538)_{10} = (7|6|2)_{RBS(9|7|4)}$
2. Негирамо те вредности: $(-538)_{10} = (-7|-6|-2)_{RBS(9|7|4)}$
3. Пребацимо нег. остатке у поз.: $(-538)_{10} = (2|1|2)_{RBS(9|7|4)}$
(адитивни инверз)

Водимо рачуна да су у истој RBS основи.

* **сабирање:** $(a_n|\dots|a_1)_{RBS(m_n|\dots|m_1)} + (b_n|\dots|b_1)_{RBS(m_n|\dots|m_1)} = (c_n|\dots|c_1)_{RBS(m_n|\dots|m_1)}$, где је $c_i = a_i +_i b_i$ ($+$ по модулу)

* **одузимање:** $(a_n|\dots|a_1)_{RBS(m_n|\dots|m_1)} - (b_n|\dots|b_1)_{RBS(m_n|\dots|m_1)} = (c_n|\dots|c_1)_{RBS(m_n|\dots|m_1)}$, где је $c_i = a_i -_i b_i$ ($-$ по модулу)

* **множење:** $(a_n|\dots|a_1)_{RBS(m_n|\dots|m_1)} \cdot (b_n|\dots|b_1)_{RBS(m_n|\dots|m_1)} = (c_n|\dots|c_1)_{RBS(m_n|\dots|m_1)}$, где је $c_i = a_i \cdot_i b_i$ ($\cdot$ по модулу)

* **дељење:** Ово је доста теже израчунати.
$(a_n|\dots|a_1)_{RBS(m_n|\dots|m_1)} / (b_n|\dots|b_1)_{RBS(m_n|\dots|m_1)} = (c_n|\dots|c_1)_{RBS(m_n|\dots|m_1)}$, где је $c_i = (a_i / b_i) \bmod i$

Међутим c_i не мора ни да постоји. Тачније постоји ако једн. $a_i \equiv b_i \cdot c_i \pmod{m_i}$ има реш.
Најлакши начин да нађемо $0 \leq c_i < m_i$ је испробавањем

* **адитивни инверз:** број $\bar{a}$ тд. $a \equiv -\bar{a} \pmod m$ (нпр. $6 \equiv -2 \pmod 8$)
у скупу реалних бр. то је његов инверз.

* **мултипликативни инверз:** број a^{-1} тд. $a \cdot a^{-1} \equiv 1 \pmod m$.
да би био деф. a и m морају бити узајамно прости.
Трази се као у дељењу: испробавамо.

6.2 Превођење бројева: показујемо из бинарног у RBS, као и из RBS у декадни.
Напомена: већ смо показали из декадног у RBS.

* **из бинарног у RBS:** $(a)_2 = (a_{n-1} \dots a_1 a_0)_2 \Rightarrow (a)_{10} = 2^{n-1} a_{n-1} + \dots + 2 \cdot a_1 + a_0$

Приметимо да је довољно да за сваки сабирак $2^i a_i$ одредимо $\bmod m_i$, а потом одредити збир тих остатака по $\bmod m_i$.

* **из RBS у декадни:** за декадне вр. записа $(a_n | \dots | a_1)_{RBS(m_n | \dots | m_1)}$ потребно је одредити тежине $t_n, \dots, t_1$

Тежина t_i је декадни бр. чији RBS запис на i-тој поз. има 1, а остало 0.
$\Rightarrow t_i \equiv 1 \pmod{m_i}$, $t_j \equiv 0 \pmod{m_j}$ за $j \neq i$ (испробавамо)

Коначно, декадна вр. броја је $\left(\sum_{i=1}^{n} a_i t_i\right) \bmod m$ $(m = m_n \cdot \dots \cdot m_1)$

6.3 Ситуација: Желимо да представимо декадне бр. из $[0, 1000000]$ преко RBS што ефикасније.
Поређења ради: у бинарном би требало да одвојимо 20 битова (јер $2^{20} > 1000000 > 2^{19}$)

Приметимо: да би се сви наведени бр. могли представити, производ модула m_i мора бити > 1000000
такође, због једнозначности, морају бити уз. прости.

* **избор 1:** првих пар простих бројева: $2 \cdot 3 \cdot 5 \cdot 7 \cdot 11 \cdot 13 \cdot 17 \cdot 19 = 9\,699\,690 > 1\,000\,000$
то је превише, па ћемо избацити 7 $\Rightarrow$ RBS(19|17|13|11|5|3|2) (и даље $> 1\,000\,000$)

број битова: $5 + 5 + 4 + 4 + 3 + 2 + 1 = 24$ (за 19 треба 5 битова итд.)

* **избор 2:** првих пар простих или степена простих: погодан је $17, 2^4, 13, 3^2, 7, 5$ $\Rightarrow$ RBS(17|16|13|9|7|5).

број битова: 23

* **избор 3:** модули облика 2^i или 2^{i-1} - погодни у рачунарима због боље искоришћене меморије.
Такође ово су компл константе код потп. и непотп. компл

нпр. RBS(2^7 | 2^7-1 | 2^5-1 | 2^2-1)

број битова: 21

6.4 Предности: лако сабирање, одузимање, множење.
велике бројеве можемо записати са мање цифара.

Мане: дељење је споро.
заузима више меморије

Примена: тамо где нема дељења
нпр. обрада дигиталних сигнала, као и телекомуникација.

7. Запис текста у рачунару.

7.1 **Коначна азбука** V је коначни непразни скуп чије елементе називамо **симболи/слова**.

Реч је коначна низа слова из V. Специјално, **празна реч** λ не садржи ни једно слово.

Скуп свих речи над V означавамо са V^*, а скуп свих непразних речи над V означавамо са V^+. $(V^+ = V^* \setminus \{\lambda\})$

Језик L је неки подскуп од V^*.

Дужина речи $|p|$ је број слова у речи p. $(|\lambda| = 0)$
p^i представља i пута дописану реч p $(p^3 = ppp)$. $p^0 = \lambda$.

пр. 1 Азбука $V = \{0,1\}$ даје језик $L = V^+$ који представља све бројеве у бин. систему (евент. водеће нуле).

7.2 **Функција кодирања** је свака $f: L_1 \to L_2$. (L_1, L_2 - језици над V_1, V_2)
Уколико постоји њена инверзна функција $g = f^{-1}$, то је **функција декодирања**.

Кодирање је израчунавање вредности $f(p)$ за неко $p \in L_1$.
Декодирање је израчунавање вредности $g(q)$ за неко $q \in L_2$.

Код $f(L_1)$ је скуп свих таквих $f(p)$. (то је слика функције f)

Особине кода: код може бити: **једнозначан** - ако је f 1-1. Иначе је **вишезначан**.
потпун - ако је f на.
равномеран - ако су све речи исте дужине

пр. 2 $V_1 = \{0,1,2,3,4,5,6,7,8,9\}$, $V_2 = \{0,1\}$. $L_1 = V_1$, а L_2 скуп свих речи V_2 дужине 4.
$f: L_1 \to L_2$ - код одговара 8421 запису

$\Rightarrow$ код је једнозначан, равномеран, али не и потпун.

7.3 Текст у рачунару је једнодимензиони низ карактера, при чему постоје и спец. карактери. (↵, EOF ...)

Сви карактери се у рачунару третирају на исти начин: сваком доделимо неки цео број на унапред деф. начин.
Кодна страна је уређена листа карактера са својим кодовима.

пр. 3 1-бајтни стандарди:

ASCII - користи 7 битова, па може 128 разних карактера.
енглеска слова, бројеви, неки интерпункцијски знакови...

ISO 8859-1 (Latin 1) - користи свих 8 битова, па може 256 разних карактера.
на првих 128 се поклапа са ASCII.
знакови западноевропских латиница.

ISO 8859-2 (Latin 2) - слично Latin 1, само овде иду источноевропске латинице.

ISO 8859-5 - слично, само овде иду источноевропске ћирилице.

пр. 4 2-бајтни стандарди:

Unicode UCS-2 - користи 2 бајта. Првих 256 се поклапа са Latin 1.
остатак карактери разних данашњих језика.
може и српска ћирилица и српска латиница

Unicode UTF-8 - сваком карактеру додељује 1-4 бајта (она која се чешће користе ⇒ мање бајтова)
може и српска ћирилица и српска латиница

8. Откривање и корекција грешака.

Грешке у рачунару се појављују при преносу података (на путу или у отпремним/пријемним уређајима).
Манифестују се тако што се један бит или низ суседних битова поремети (заменимо 0 и 1).

Два начина за откривање: 1. **контрола грешке уназад** - може само да се установи има ли грешке.
ако је дошло до грешке, шаље се још једна порука
уз битове поруке, шаљу се и додатни битови (само за присуство)
- порука заузима мало ⇒ погодно за између 2 уређаја.

2. **контрола грешке унапред** - и откривање и корекција.
ако је дошло до грешке, шаље се (још једна) порука + (корекција бајтова) уз битове поруке, шаљу се и додатни битови (присуство + где је)
- порука заузима више ⇒ погодно за 1 уређај.

Типови грешака: * појединачан тип: један бит
* проширен тип: низ суседних битова

Bits error rate, **BER**, је вероватноћа да један бит у одређеном временском интервалу има грешку.

пр. 1 $BER = 10^{-6}$ ⇒ у просеку 1 од 10^6 битова у одређеном временском интервалу има грешку.

Наводимо прво 3 алгоритма за откривање, а онда један и за откривање и за корекцију.

8.1 **Контрола парности:** на крају поруке додамо 1 ако је бр. јединица непаран, 0 ако је паран.

пр. 2 Шаљемо поруку „101101" и додамо 0 (јер је бр. јединица 4) ⇒ шаљемо 1011010.
Међутим, дошло је до грешке и први бит је постао 0. Дакле, примљено је 0011010.
Али тада има непаран бр. јединица, па прималац може да установи само постојање грешке.

8.2 **Контрола збира блока:** уз саму поруку шаљемо и додатно бр. јединица (записан бинарно)

пр. 3 Шаљемо поруку „101101" и додамо 4 (јер је бр. јединица 4) ⇒ шаљемо 101101|4 („100").
Међутим, дошло је до грешке и 3. и 4. бит су постали 0. Дакле, примљено је 100001|4
Али тада има мањи број јединица, па прималац може да установи само постојање грешке.

Циклична провера редунданци / CRC метода: уз кодирану поруку шаљемо и **полином генератор**, $G(x)$.
(полином чији су коеф. 0 или 1)

пр. 4 * Желимо да пошаљемо поруку „11100110".

Изаберемо $G(x) = x^4 + x^3 + 1$, који запишемо као 11001. $(1 \cdot x^4 + 1 \cdot x^3 + 0 \cdot x^2 + 0 \cdot x + 1)$

На поруку додамо $\deg G$ (4) нула: 111001100000.

Такав бр. „поделимо полиномом", али гледамо само остатак (110). Њега допунимо до $\deg G$ цифара (0110).
↳ уместо одузимања је ексклузивна дисј. (нема позајмица)

На почетну поруку додамо остатак и шаљемо то (111001100110) и полином генератор. $(G(x) = x^4 + x^3 + 1)$

* Сада смо у улози примаоца: њему је стигла нека порука 11001011101 и $G(x) = x^2 + 1$.
Запис полинома је 101.
Подели се цела примљена порука са полиномом: 1° остатак није 0: грешка (ми добијамо 11)
2° остатак јесте 0: нема грешке
⇒ одбацимо $\deg G$ цифара са краја ⇒ порука.

8.4 **Хамингов SEC код:** може и да изврши корекцију грешке на биту на ком се појавила.

пр. 5 Кодирана порука која је послата је: 101001100110.
Порука има 12 битова: 8 је сама порука, 4 су контролни битови.

m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	c_1	c_2	c_3	c_4
1	0	1	0	0	1	1	0	0	1	1	0

Прималац, на основу примљене поруке генерише своје контролне битове c_i'.
Тако закључује да ли је дошло до грешке на неком биту и (евентуално) изврши корекције.

Формирамо **Хамингову таблицу**. c_i' = екскл. дисј. оних m_i који у i-тој колони бин. записа имају 1.

12	(1)100	m_8
11	(1)011	m_7
10	(1)010	m_6
9	(1)001	m_5
8	1000	c_4
7	0111	m_4
6	0110	m_3
5	0101	m_2
4	0100	c_3
3	0011	m_1
2	0010	c_2
1	0001	c_1

c_i - степени двојке
m_i - редом

нпр. $c_4' = m_5 \oplus m_6 \oplus m_7 \oplus m_8 = 0 \oplus 1 \oplus 0 \oplus 1 = 1 \oplus 0 \oplus 1 = 1 \oplus 1 = 0$

Слично: $c_3' = 1$, $c_2' = 0$, $c_1' = 0$.

Рачунамо $c_4 c_3 c_2 c_1 \oplus c_4' c_3' c_2' c_1' = 0110 \oplus 0101 = 0011$

Нађемо тај број у табели ⇒ ту је грешка ⇒ инвертујемо тај бит

0011 → m_1 ⇒ грешка у m_1 ⇒ исправна порука је 101001100110

Ако тај број пада на c_i или га уопште нема ⇒ нема грешке.

9. Историјат рачунарских система.

9.1 **Рачунарско средство** је свако помагало које је израђено у циљу извршавања рачунских операција.

↳ подела по начину извршавања операција:
- *ручно - лењир
- *полуаутоматско - калкулатор
- *аутоматско - рачунар

↳ подела по физичким принципима на којима се изводи операција:

*континуална:
- њихов математички модел је еквивалентан математичком моделу проблема.
- подаци се представљају преко непрекидних физичких величина.
  ↳ прецизност зависи од квалитета израде
- шибер (клизајући лењир) и аналогни рачунари (1960', улаз = волтажа).

*дискретна:
- обављају операције са дискретним подацима.
- подаци се представљају преко бројева.
  ↳ прецизност не зависи од квалитета израде.
- абакус и савремени рачунари.

9.2 **Периоди у развоју информационих технологија:**

i) **премеханички:** писма и бројеви. Абакус.

- Сумери (3000 п.н.е.) - клинасто писмо
- Феничани (2000 п.н.е.) - први алфабет, састављен је од слогова и сугласника
- Грци - додали самогласнике, Римљани - доделили латинска имена словима.

- Египћани (2600 п.н.е.) - писали на папирусу, Кинези (100 н.е.) - папир
- прве библиотеке - Грчка, 500 п.н.е.

- нелозиционни бројевни системи (египатски, римски)
- Индуси - први позициони бројевни систем (9 цифара), Арапи - додали нулу

ii) **механички:** почиње кад је Јохан Гутенберг открио штампарску пресу.

* Шибер: клизајући лењир.
* Шикардова машина: +/- за 6-цифрене, аларм за прекорачење.
* Паскалова машина: +/- за 8-цифрене.
* Лајбницова машина: +/-/·/÷
* аритмометар: +/-/·/÷, први пут серијска производња.
* комптометар: први пут унос преко тастера.
* аналитичка машина, Чарлс Бебиџ: „претеча" данашњих (имала меморију) никад није довршена.

iii) **електромеханички:** почиње кад су информације почеле да се конвертују у електричне импулсе.
телеграф, телефон, радио

* **Холеритов табулатор:** за мало времена пуно података. Пописи становништва у САД. (XIX-XX в.)
* **Z1:** први електромеханички калкулатор.
* **MARK I:** први електромеханички програмибилни калкулатор. (реални бр. у фикс. зарезу)
* **Enigma:** шифровање и дешифровање, у Немачкој.

iv) **електронски:** пред II светски рат.
Теслино електронско логичко коло ⇒ ел. вакуумске цеви и флип-флоп ел. кола.
телевизија и екран са катодном цеви.

* **ABC** - решавање линеарних једначина.
* **Колос** - дешифровање Немачких порука (вакуумске цеви).
* **ENIAC** - 30 тона, декадни систем, основне рачунске операције
* **EDSAC** - први оперативни рачунар који чува програме у меморији.
* **BINAC** - први рачунар са дуалним процесором.
* **Вихор** - први рачунар за рад у реалном времену (500 000 сабирања у секунди).

9.3 **Генерације рачунара** - подела по технологијама које су коришћене за израду рачунара.

I генерација - крај '30 → крај '50
- логички елементи: **вакуумске цеви** - непоуздане, троше много енергије, загревају се, велике
  унутрашња меморија: **магнетне траке** и **магнетни добоши**
  улазно-излазни уређаји: **бушене картице**, писаће машине, штампачи
- програмирање: **машински**, после **асемблер**, на крају **Fortran**.
- **UNIVAC**

II генерација - крај '50 → средина '60

- логички елементи: **транзистори** - много практичније и јефтиније.
  унутрашња меморија: **магнетна језгра** - чувају и кад је угашен.
- програмирање: **Cobol**, **Lisp**, први **оперативни системи**.

III генерација - средина '60 → средина '70

- логички елементи: **интегрисано коло** - више спојених транзистора ⇒ рачунари мањи
- програмирање: још **оперативних система**, **Pascal**, **C** ⇒ комерцијализација
- **PDP-8**, **серија S/360** (IBM)
  ↳ први минирачунар на тржишту ↳ компатибилни међусобно

IV генерација - средина '70 → данас
- логички елементи: **минијатуризација интегрисаних кола**
  унутрашња меморија: **полупроводничка**
- **персонални рачунари** (нпр. MITS Altair 8800)