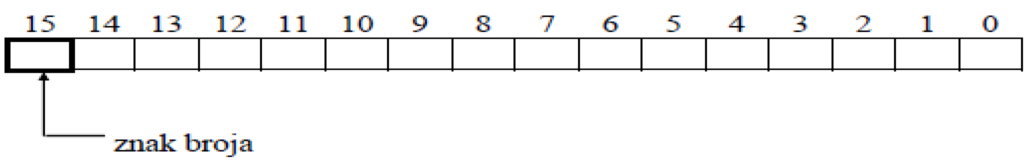
ZAPIS BROJEVA U RAČUNARU

Pamćenje celih brojeva u mikroračunarima i računske operacije sa njima

- Za pamćenje označenih celih brojeva (integer) u mikroračunarima u normalnom radu se angažuju (pri obradi programa napisanih u višim programskim jezicima) **dva bajta** odnosno dve memorijske lokacije.



- Bitove u dvobajtnom broju označavaćemo sa b0 - b15. Pri tome, bitovi b0 - b7 pripadaju manje značajnom bajtu (*Low byte*), a bitovi b8 - b15 značajnijem bajtu (*High byte*).
- Najznačajniji bit b15 čuva informaciju o znaku broja:

$$b_{15} = \begin{matrix} 0 \\ 1 \end{matrix} \left. \right\} \begin{matrix} \text{za znak +} \\ \text{za znak -} \end{matrix}$$

- Tako se označeni celi celi brojevi (integer) pamte (kodiraju) u vidu 2 – komplementa.

- Primer 1
- Prikaži sadržaj memorijskih lokacija u kojima su smešteni brojevi
- $x = 1285_{10}$; $y= -1285_{10} = -x$.
- binarni oblik broja x: $1285_{10} = 10100000101_2$
- U dva bajta memorije: x=0000010100000101
- ili skraćeno uz pomoć heksadekadnog prikaza: $x=0505_{16}$
- Za y= -x imamo:
- y=(0000010100000101)c
- y=1111101011111010=1111101011111011

  +1
- Rezultat je praktičnije tražiti u heksadekadnom sistemu:
- $y=(0505_{16})c=FAFA = FAFB_{16}$

  +1

- Kako je kapacitet memorijske lokacije ograničen, postavlja se pitanje koliki je najveći, a koliki najmanji ceo broj koji se može zapamtiti (registrovati).
- Najveći ceo broj x_{max} biće:
- x_{max}=0111111111111111 = 1000000000000000 - 1
- x_{max}= 2^{15} - 1= 32768 - 1 = 32767
- To je relativno mali broj i ako se u nekom problemu pojavljuju veći celi brojevi, kako prevazići to ograničenje ?
- U višim programskim jezicima (BASIC, FORTRAN, PASCAL, C) postoje komande koje kao rezultat imaju angažovanje **duplo duže lokacije** (4 bajta) za registrovanje celih brojeva (*long integer*). Tada se gornja granica pomera na:
- x_{max}= 2^{31} - 1 = 2147480000
- što može biti dovoljno.

- Ukoliko i to ne rešava problem, preostaje da se celi brojevi u računaru obrađuju kao realni, čime se značajno povećava gornja granica. To se međutim, kao što ćemo videti, mora “platiti” gubitkom tačnosti u toku računskih operacija.
- Najmanji ceo broj koji se može registrovati u dvobajtnoj lokaciji, x_{min} (veliki po apsolutnoj vrednosti negativan broj) nije jednak $-x_{max}$ iz razloga što se negativni brojevi registruju u vidu 2 - komplementa. Tako broj
- $x=1111111111111111=FFFF_{16}$
- nije, što se može u prvi mah pomisliti, traženi najmanji broj, on ustvari predstavlja broj 1 jer je:
- $(FFFF)c=0001_{16}$

- Najmanji broj u računaru ima oblik:
- $1000000000000000 = 8000_{16}$
- pa je:
- $x_{min} = -\ (8000_{16})c = -7FFF = -8000_{16}$

  $+1$
- $x_{min} = -8\ 16^3 = -2^3\ 2^{12} = -2^{15}$
- $x_{min} = -32768$

- Što se računskih operacija tiče, one se svode na sabiranje kao što smo to u prethodnim poglavljima videli. Deljenje, kao što znamo, nije definisano na skupu celih brojeva i po pravilu se u računaru kao rezultat delenja dva cela broja, tzv. celobrojno deljenje, uzima ceo broj koji predstavlja celobrojni deo tačnog rezultata.

Primer 2

- Šta će se u računaru sa jednobajtnom memorijskom lokacijom dobiti pri izvođenju sledećih operacija:

$$j = a \cdot b\,, \qquad k = b : a$$

gde su $a=24_{10}$; $b=49_{10}$.

- Najveći označen ceo broj u 1-bajtnoj lokaciji će biti:
- $x_{max}=2^7-1 = 127$
- Kako je: $24 \cdot 49 > 127$

doći će do aritmetičkog prelivanja, odnosno računar neće moći da dobije rezultat, o čemu korisnik dobija odgovarajuće upozorenje (*aritmetic overflow*).

- Pri delenju, 00110001:00011000 postupak se prekida, pošto se dobija celobrojni deo rezultata:

```
  110001 : 11000 = 10
- 11000
  000001
```

- Dakle rezultat će biti:
- $k=10_2 = 2_{10}$
- što predstavlja celobrojni deo tačnog rezultata, 2.041666...

Primer 3

- a) Prikazati sadržaj 16-bitnih memorijskih lokacija u koje su smešteni brojevi:

  $n = -FAD_{16}$ i $m = 2756_{10}$
- b) Izračunati: $k_8 = n_8 + m_8$ uz pomoć 8-komplementa
- c) Prikaži sadržaj lokacije u kojoj je smešten rezultat delenja: $h = n / m$

- a) $n = -FAD_{16} = -111110101101_2$
- U memoriji:

$$n = (0000111110101101)_c = (0FAD_{16})_c = F052 = F053_{16} = 1111000001010011$$

$$+1$$

- Da bi smo prikazali izgled broja m u računaru, prevešćemo ga najpre u oktalni oblik koji nam treba u problemu pod b

  $m = 5304_8 = 101011000100_2$
- U memorijskoj lokaciji $m = 0000101011000100 = 0AC4_{16}$

2756 : 8	
344	4
43	0
5 →	3

- b) $n = -111110101101_2 = -7655_8$

  $m = 5304_8$
- Pošto oba broja imaju po 4 cifre, pri primeni komplementa moramo da radimo sa peto ili višecifrenim brojevima:

$m = 05304_8 \quad n = -07655_8$

- $k = m + n = m - |n| = m + |n|_c$
- $|n|_c = (07655)_c = 70122 = 70123$

$$\begin{array}{r} k=70123 \\ \underline{+05304} \\ 75427 \end{array} \qquad \text{(k<0 jer je prva cifra B-1=7)}$$

$k = -(75427)_c = -02350 = -2351_8$

c) Pošto je $n<0$:

$$n/m = -(|n|/m)$$

$$|n|:m = 111110101101 : 101011000100 = 1 \ldots.$$

Računar dobija samo celobrojni deo rezultata, pa je:

$$h = -1$$

u memoriji:

$$h = (0001_{16})_c = \mathrm{FFFE} = \mathrm{FFFF}_{16} = 1111111111111111$$

$$+1$$

Primer 4

- U bajtovski organizovanoj memoriji (memorijska lokacija koja ima svoju adresu je 1 bajt) u nizu susednih lokacija koji počinje na adresi $0FFF_{16}$, a završava na adresi $10FA_{16}$, smešteni su redom elementi nekog celobrojnog niza: xi, i=1,n.
- a) Koliko je dugačak niz?
- b) Ako je izgled posmatrane zone u memoriji (heksadekadni prikaz):

adresa:

$$\overbrace{10}^{0FFF_{16}}\ 3A07BCE50F68E90579A0C\ldots$$

- Kako će se promeniti sadržaj posmatranih lokacija posle izvršenja operacija:
- $x_5=x_1-x_3$
- $x_4=x_3/x_2$

- a) Ukupan broj bajtova u posmatranoj memorijskoj zoni dobićemo kada od adrese poslednjeg bajta oduzmemo adresu prvog i tome dodamo 1:

broj bajtova = (10FA-0FFF)+1

broj bajtova= FC_{16} = 15 · 16+12=252

$$\begin{array}{r} 10FA \\ -\ \underline{0FFF} \\ 00FB \end{array}$$

- Pošto se za svaki od članova niza, kao ceo broj angažuje dva bajta, broj članova niza biće:

n = 252/2 = 126

b) U prikazanom nizu bajtova identifikujemo prvi i treći član niza.

$x_1=103A_{16}=\boxed{0}\,001000000111010;$ $x_1>0$

$x_3=E50F_{16}=\boxed{1}\,110010100001111;$ $x_3<0$

$x_5=x_1-x_3=x_1+(x_3)_c$

- Operacije ćemo brže izvesti u heksadekadnom sistemu:

- $(x_3)_c=(E50F_{16})_c=1AF0=1AF1$

  $+1$

$$\begin{array}{r} x_5=103A \\ \underline{+1AF1} \\ 2B2B \end{array} \quad (x_5>0)$$

- Pošto je x_3 negativan, deli se $|x_3|$ sa $x_2=07BC_{16}$ $(x_2>0)$:
- $|x_3|=-x_3=(E50F_{16})_c=1AF1=1101011110001_2$
- $x_2=11110111100_2$
- Deljenje se prekida pošto je izračunat celobrojni deo rezultata:

$$\begin{array}{l} 1101011110001 : 11110111100 = 11 \\ -11110111100 \\ \quad 101101111001 \\ -\ 11110111100 \\ \quad\quad 1110111101 \end{array}$$

- $x_3/x_2=-11_2=-3_{16}$
- $x_4=-0003_{16}=(0003_{16})_c=FFFC=FFFD_{16}$

  $+1$

- Po izvršenju operacija promeniće se sadržaji bajtova u kojima su smešteni elementi x_4 i x_5 (ukupno 4 bajta). Novi sadržaj:

$$\overbrace{FF}^{\substack{\text{adresa=} \\ 0FFF+3\cdot 2}} \quad FD2B \quad \underbrace{2B}_{\substack{\text{adresa=} \\ 1008_{16}}}$$

Realni brojevi u računaru

- Da se podsetimo najpre da se u dekadnom sistemu realni brojevi mogu zapisati u dva oblika: oblik sa **nepokretnom decimalnom tačkom** (*fixed point number*) i **eksponencijalni** oblik ili oblik sa **pokretnom decimalnom tačkom** (*floating point number*).
- Na primer:

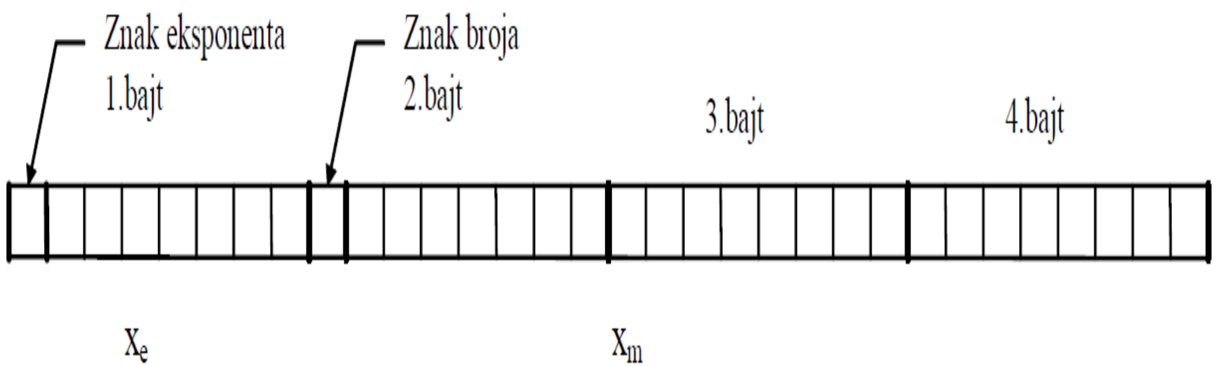
$$\underbrace{0.0257}_{\substack{\text{oblik sa fiksnom}\\ \text{dec. tackom}}} = \underbrace{2.57\cdot 10^{-2}}_{\substack{\text{oblik sa pokretnom}\\ \text{dec. tackom}}} = 0.0257\cdot 10^{0} = 0.00257\cdot 10^{1} = \ldots$$

- Jedan od bekonačno mnogo oblika sa pokretnom decimalnom tačkom, usvojen je kao standardni i zove se **normalizovani eksponencijalni oblik**.
- U posmatrnom primeru to je: $0.257 \cdot 10^{-1}$
- gde se predeksponencijalni faktor 0.257, koji predstavlja pravi razlomak čija je prva decimala različita od nule, naziva **mantisa**, a stepen osnove, -1 naziva se **eksponent.**

- Uopšte, u nekom pozicionom brojnom sistemu sa osnovom B, normalizovana eksponencijalna forma broja x je:
- $x = \pm x_m \cdot B^{xe}$ (8)
- $0.1 \leq x_m < 1$ (8a)
- gde x_m označava mantisu, a x_e eksponent.
- Na primer:
- $-1011.01_2 = -0.101101_2\ \mathbf{2}^{100}$; $x_m = 0.101101_2$; $x_e = 100_2$
- $0.00731_8 = 0.731_8\ \mathbf{8}^{-2}$; $x_m = 0.731_8$; $x_e = -2_8$
- U oba primera smo baze sistema 2 i 8 ostavili u dekadnom obliku umesto da ih prikažemo u odgovarajućem brojnom sistemu: $2=10_2$; $8=10_8$ da bi bilo uočljivije o kojoj se osnovi radi.
- Naime, u bilo kom brojnom sistemu, osnova sistema kao broj, ima isti kod:
- $B=10_B$
- $10=10_{10}$; $2=10_2$; $8=10_8$; $16=10_{16}$ . . .

- Važan pojam vezan za realne brojeve je **broj značajnih cifara** u broju, i on je u direktnoj vezi sa **tačnošću informacije koju sadrži posmatrani broj.** Neka smo, na primer, na analitičkoj vagi, za koju znamo da ne može da registruje mase manje od 10^{-4} g, izmerili 1 g neke supstance. Za masu supstance x pisaćemo: x=1.0000 g, a ne x=1 g, da bi naglasili tačnost informacije koju vaga daje. Ako smo pak izmerili 0.0205 g, nećemo to pisati kao 0.02050, već samo sa četiri decimale, y=0.0205 g jer nula na petoj decimali nije rezultat merenja. Sve cifre u broju x su značajne cifre (ukupno 5), dok broj y ima samo 3 značajne cifre, poslednje tri. Naime, **nule na početku broja** (“leve“ nule) **ne predstavljaju značajne cifre**, **dok su nule na kraju broja** (“desne” nule) **značajne**, **kao i sve nule između dve značajne cifre.**
- Zašto “leve“ nule nisu značajne biće jasno ako masu 0.0205 g prikažemo u različitim jedinicama:
- 0.0205 g=20.5 mg=0.0000205 kg . . .
- Očigledno, njihov broj varira zavisno od odabrane jedinice mere, te ne predstavlja informaciju o tačnosti merenja (koja je nezavisna od odabrane jedinice mere).
- Treba zapaziti da prema datom pravilu, u normalizovanom eksponencijalnom broju, *sve cifre decimalnog dela mantise predstavljaju značajne cifre.*
- Na primer: y=0.205 10^{-1} g=0.205 10^{2} mg=0.205 10^{-4} kg

- U kompjuterskoj aritmetici, broj značajnih cifara daje informaciju o tačnosti sa kojom je neki realan broj registrovan u memoriji.
- Realni brojevi se u memoriji računara čuvaju u normalizovanom eksponencijalnom binarnom obliku. Dakle memorijski prostor za neki realan broj ima dva dela: **deo za mantisu i deo za eksponent.** Standardna organizacija registrovanja realnog broja data je na slici



- Realan broj u memoriji zauzima četiri bajta, odnosno kod mikroračunara to su četiri memorijske lokacije. Prvi bajt služi za pamćenje eksponenata i za njega važe pravila za pamćenje celih brojeva: *Najznačajniji bit predstavlja znak eksponenta, a negativni eksponenti se čuvaju u obliku 2-komplementa.*

- **Prvi (najznačajniji), od ukupno 24 bita koliko je rezervisano za mantisu, čuva informaciju o znaku broja, a ostali predstavljaju decimale mantise.** Pri tome treba imati u vidu na osnovu definicije normalizovane eksponencijalne forme (8, 8a) da ako je najznačajniji bit jednak nuli (pozitivna mantisa), onda bit iza najznačajnijeg mora biti jednak jedinici.
- Za negativne brojeve u prostoru za mantisu smešta se 2 - komplement mantise.
- Uporedimo sada interne ili mašinske kodove brojeva:
- -4 , ceo broj i -4.0 , realan broj
- Interni kod (oblik u kome je broj u memoriji) celog broja -4 dugačak je 16 bitova (2 memorijske lokacije):
- $-4 = -100_2 = -0000000000000100 = -0004_{16}$
- $-4 \rightarrow (0004_{16})_c = FFFC_{16} = 1111111111111100$
- Da bi smo odredili izgled realnog broja -4.0 u memoriji, prikazaćemo ga u normalizovanom eksponencijalnom binarnom obliku.
- $-4.0 = -100.0 = -0.1 \cdot 2^{11}$
- Prvi bajt internog koda je eksponent: $x_e = 00000011 = 03_{16}$

- Tri bajta predstavljaju mantisu x_m, a pošto je broj negativan:

$$x_m = \left(01\underbrace{00000\ldots0}_{22\ \text{nule}}\right)_c = (400000_{16})_c = BFFFFF = C00000_{16}$$

- Dakle, izgled broja -4. u memoriji je:

$$-4. \rightarrow \underbrace{03}_{x_e}\underbrace{C00000}_{x_m}{}_{16} = \underbrace{00000011}_{x_e}\underbrace{110\ldots0}_{x_m}$$

- Primer 5
- a) Brojeve $x = -E1C.07_{16}$ i $y = 1287.59_{10}$ prevesti u oktalne sa tačnošću od 4 decimale i potom ih sabrati.
- b) Prikazati izgled rezultata u memoriji računara.

- a) Prvi broj ćemo prevesti u oktalni posredstvom binarnog oblika:
- $E1C.07_{16} \rightarrow 111000011100.00000111_2 \rightarrow 7034.016_8$
- $x = -7034.016_8$
- Za drugi broj posebno konvertujemo celobrojni, a posebno decimalni deo.

$$\begin{array}{ll} \multicolumn{2}{l}{\underline{1287:8}} \\ 160 & 7 \\ 20 & 0 \\ 2 & 4 \end{array} \qquad 1287_{10}=2407_8$$

$$\begin{array}{ll} \underline{0.} & \underline{59} \ \mathrm{x}\,8 \\ 4. & 72 \\ 5. & 76 \\ 6. & 08 \\ 0. & 64 \\ 5. & 12 \end{array}$$

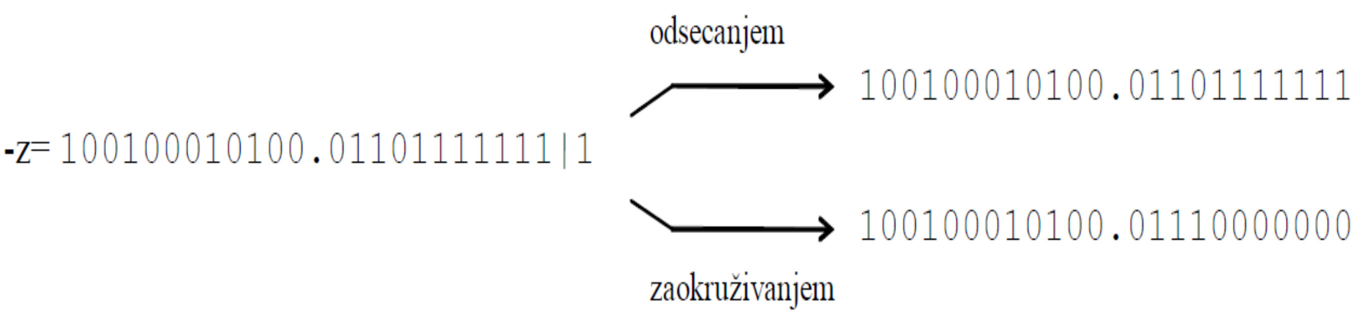
- Kako je peta decimala 5, veća od polovine osnove, po pravilima zaokruživanja: $0.59_{10}=0.4561_8$
- $y=2407.4561_8$
- x+y=-7034.016+2407.4561
- Oduzimamo manji od većeg broja
- Rezultat je:
- z=x+y=-4424.33778

$$\begin{array}{r} 7034.016 \\ -\ \underline{2407.4561} \\ 4424.3377_8 \end{array}$$

- b) -z=4424.3377=100100010100.011011111111
- Broj ima ukupno 8x3=24 binarne značajne cifre. U prostoru od 3 bajta,

predviđenom za znak broja (prvi bit) i decimale mantise (preostala 23 bita) ima mesta za 23 binarne značajne cifre, tj. broj z se nemože tačno prikazati u memoriji. Neophodno je zameniti ga 23-bitnim brojem, i to se može izvesti:

- - **odsecanjem**, tj. jednostavnim odbacivanjem viška bitova
- - **zaokruživanjem** na 23 značajne cifre, čime se smanjuje greška



- Pri zaokruživanju, pošto je najznačajniji odbačeni bit jednak 1, (polovina osnove) na poslednju značajnu cifru dodaje se 1.

- Iz oblika sa fiksnom decimalnom tačkom prelazimo u normalizovani eksponencijalni oblik, da bi odredili mantisu i eksponent.

$$z = \begin{cases} -0.1001000101000110111111 \cdot 2^{1100} & \text{(odsecanje)} \\ -0.1001000101000111000000 \cdot 2^{1100} & \text{(zaokruzivanje)} \end{cases}$$

- $z_e = 1100 = 00001100 = 0C_{16}$
- Pošto je broj negativan, u prostoru za mantisu smešta se 2-komplement mantise, koga možemo odrediti uz pomoć heksadekadnog koda:

$$(z_m)_c \begin{cases} (48A37F_{16})_c = B75C81_{16} & \text{(odsecanje)} \\ (48A380_{16})_c = B75C80_{16} & \text{(zaokruzivanje)} \end{cases}$$

- Konačno, izgled broja z u memoriji je u heksadekadnom prikazu:

$$z: \begin{cases} 0CB75C81 & \text{(odsecanje)} \\ 0CB75C80 & \text{(zaokruzivanje)} \end{cases}$$

Primer 6

- Dati su sadržaji memorijskih lokacija u kojima su smešteni realni brojevi x i y.
- x: 04A16900   y: FE560000
- a) Prikazati brojeve x i y u heksadekadnom brojnom sistemu u obliku sa fiksnom
- decimalnom tačkom.
- b) Izračunati njihov zbir.
- a) $x_e=04_{16}=4_{10}$ x_m : A16900
- U pitanju je komplement mantise jer je prvi bit jednak jedinici. Nalazimo mantisu:
- $(A16900_{16})c=5E96FF=5E9700_{16}$
-                                 +1
- $x_m=0.101111010010111$
- Pošto je broj negativan:
- $x=-\ x_m \cdot 2^{xe} = -\ 1011.11010010111_2 = -\ \ B.D2E_{16}$
- y_e : FE (u pitanju je negativan eksponent, u obliku komplementa)
- $y_e=-(FE_{16})_c=-02_{16}=-2_{10}$
- y_m : 560000, $y_m=0.101011$

$$y = y_m \cdot 2^{y_e} = 0.101011 \cdot 2^{-2} = 0.00101011 = 0.2B_{16}$$

- b) x+y=-B.A7E16

$$\begin{array}{r} B.D2E \\ -\ 0.2B \\ \hline B.A7E \end{array}$$

- Primer 7
- Heksadekadni prikaz sadržaja dela memorije u kome je smešten niz realnih brojeva je: $\underbrace{11}_{\text{adresa: } 101B_{16}}$ 4E2930213AC000109DFB68A.....C $\underbrace{01}_{\text{adresa: } 121E_{16}}$
- a) Koliko niz ima članova?
- b) Koji bajtovi i kako se menjaju nakon operacije: x2=x1+x3
- a) Ukupan broj bajtova u memorijskoj zoni u koju je smešten realan niz je;
- br. bajtova=$121E_{16}$ - $101B_{16}$+1=204_{16}
- br. bajtova=2·256+4=516
- Kako svaki član niza zauzima po 4 bajta, broj članova niza je:
- n=br. bajtova / 4=129

- b) Identifikovaćemo članove niza x_1 i x_3:
- x_1 : 114E2930
- $x_e=11_{16}=17_{10}$
- x_m : 4E2930
- x_m=0.10011100010100100110000

$$x_1 = x_m \cdot 2^{x_e} = 10011100010100100.11_2 = 138A4.C_{16}$$

- x_3 : 109DFB68
- $x_e=10_{16}=16_{10}$
- x_m : 9DFB68 (u pitanju je komplement)
- (9DFB6816)c=62049816
- x_m=0.11000100000010010011000

$$x_3 = -x_m \cdot 2^{x_e} = -1100010000001001.0011$$

$x_3=-C409.3_{16}$

- Računamo x2:

$$\begin{array}{r} 138A4.C \\ -\ C409.3 \\ \hline 749B.9 \end{array}$$

- $x2=749B.9_{16}=111010010011011.1001_2$
- Nalazimo eksponent i mantisu:
- $x_e=15=0F_{16}$
- $x_m=0.111010010011011 1001$
- x_m : 749B90
- Kako x_2 zauzima 4 bajta na adresama:
- $(101B+4)_{16}$ do $(101B+7)_{16}$
- novi sadržaj tih bajtova je:

$$\underbrace{0F}_{\substack{\text{adresa:}\\ 101F}}\ 749B90$$

Ograničenja pri registrovanju realnih brojeva

- S obzirom na ograničenu dužinu (broj bitova) memorijskog prostora za eksponent, postoji gornja granica veličine realnog broja koji se može registrovati, x_{max}. Naćićemo je kao: $x_{max} = (x_m)_{max} \cdot 2^{(x_e)_{max}}$
- Prikaz maksimalne mantise $(x_m)_{max}$ u računaru je: $0\underbrace{111\ldots1}_{23}$
- To je $0.\underbrace{111\ldots1}_{23} = 1 - 0.\underbrace{00\ldots1}_{23} = 1 - 2^{-23}$
- Maksimalan eksponent je: $(x_e)_{max} = 2^7 - 1 = 127$
- pa imamo: $x_{max} = (1 - 2^{-23})\, 2^{127} \approx 1.708\ 10^{38} \approx 10^{38}$
- Analognim postupkom za najmanji realan broj, koji se može registrovati, dobijamo:

$$x_{min} = -x_{max} \approx -1.708\ 10^{38} \approx -10^{38}$$

- *Interne vrednosti realnih brojeva koje su po apsolutnoj vrednosti veći od granice:* $|x|>x_{max}$ *biće jednake gornjoj granici:* $x = x_{max}$ (*aritmetic overflow*).

- Dužina eksponenta uslovljava takođe i **donju granicu apsolutne vrednosti broja** ispod koje su svi brojevi za računar jednaki nuli. Ovo ograničenje bi mogli nazvati ***ograničenje preciznosti registrovanja***.

$$|x|_{min}=(x_m)_{min}\cdot 2^{(x_e)_{min}}$$
$$(x_m)_{min}=0.1$$
$$(x_e)_{min}=-2^7=-128$$
$$|x|_{min}=2^{-1}\,2^{-128}=2^{-129}\approx 1.46\;10^{-39}\approx 10^{-39}$$

- Dakle, ako je :$|x|< x_{min}$ interna vrednost broja x biće jednaka nuli x=0 (*aritmetic underflow*).

- Konačno, treće ograničenje - **ograničena tačnost registrovanja**, posledica je ograničene dužine mantise u memoriji. Tačnost registrovanja realnih brojeva tražićemo u vidu broja značajnih dekadnih cifara, koje se mogu registrovati.
- Najpre možemo da odredimo broj značajnih binarnih cifara - on je jednak broju bitova u prikazu decimala mantise: 23.
- Dakle, vrednost najmanje značajne cifre koja se može registrovati je 2^{-23}
- Kako je:
- $10^{-7} < 2^{-23} < 10^{-6}$
- znači da je poslednja dekadna cifra mantise, koja se može tačno registrovati približno 10^{-7}, pa je približno, *tačnost prikazivanja realnih brojeva jednaka 7 značajnih cifara*.

Računske operacije sa realnim brojevima

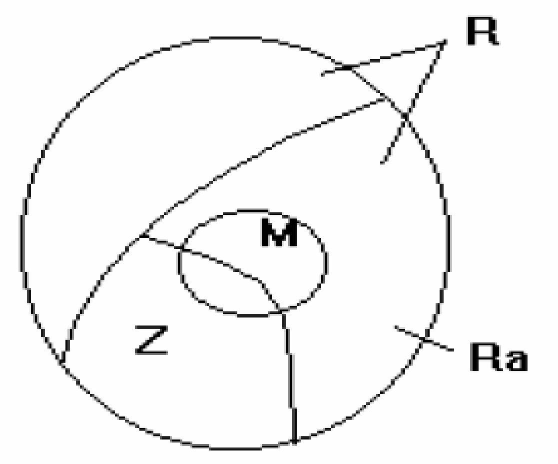
- U pitanju su operacije sa brojevima u eksponencijalnom obliku koji se sastoje od odgovarajućih operacija između predeksponencijalnih faktora i eksponenata operanada.
- Tako se **sabiranje** dva realna broja obavlja u sledećim koracima:
- 1. Dovođenje brojeva na isti eksponent **denormalizacijom** predeksponencijalnog faktora (mantise) manjeg od brojeva
- 2. Sabiranje predeksponencijalnog faktora jednog i mantise drugog broja
- 3. Normalizacija, tj. dovođenje predeksponencijalnog faktora zbira u interval [0.1,1), odgovarajućim korigovanjem eksponenta
- **Oduzimanje** se svodi na sabiranje, tj. sabira se predeksponencijalni faktor prvog broja sa komplementom predeksponencijalnog faktora drugog broja.
- **Množenje** uključuje:
- 1. Množenje mantisa
- 2. Sabiranje eksponenata
- 3. Normalizacija rezultata i definisanje njegovog predznaka

- **Deljenje** uključuje:
- 1. Deljenje mantisa
- 2. Oduzimanje eksponenata
- 3. Normalizacija rezultata i definisanjenjegovog predznaka
- Za realizaciju operacija sa realnom brojevima neophodno je raspolagati sklopovima za operacije sabiranja i komplementiranja celih brojeva, pomeranja (*shift*) i inkrementiranja i dekrementiranja (povećanja ili smanjenja vrednosti celog broja za 1).
- Zbog različitog načina registrovanja u memoriji *realni i celobrojni operandi su međusobno imkompatibilni*, *tj*. CPU *ne može da izvede operaciju između realnog i celobrojnog operanda*.
- Zato u višim programskim jezicima postoje funkcije (grupe instrukcija) za prevođenje celog broja u realan oblik, kojima se *aktivira odgovarajući niz mašinskih instrukcija za transformaciju nekog* ***dvobajtnog celobrojnog*** *operanda u* ***četvorobajtni realni***.

Ograničenje kompjuterske aritmetike

- Kako je za predstavljanje brojeva u računaru na raspolaganju ograničena memorijska lokacija, jedan ograničen podskup skupa realnih brojeva se može tačno predstaviti (kodirati).
- Internu vrednost tj. kod nekog broja, zvaćemo **mašinski broj**. Ograničenja pri predstavljanju realnih brojeva u računaru, tj. preslikavanju skupa realnih brojeva **R** u skup mašinskih brojeva **M,** jasna su sa skice

- *Odnos skupova* **R** i **M**



- R - skup realnih brojeva
- Ra - skup racionalnih brojeva
- Z - skup celih brojeva
- M - skup mašinskih brojeva

- *Preslikavanje iz skupa realnih u skup mašinskih brojeva*: γ : **R** → **M** *naziva se* ***redukciono preslikavanje.***
- Tako, interni kod broja x možemo da označimo kao γx, gde je γ operator redukcionog preslikavanja.
- Ako najmanji pozitivan realan broj koji se može registrovati u računaru označimo sa xmin, $x_{min}=|x|_{min}$, a najveći pozitivan realan broj sa xmax, skup **R** se može raščlaniti na sledeće podskupove:
- $\mathbf{R}=\mathbf{R}_{-\infty} \cup \mathbf{R}_{-1} \cup \mathbf{R}_{\mathbf{0}} \cup \mathbf{R}_{1} \cup \mathbf{R}_{\infty}$
- $\mathbf{R}_{-\infty} = (-\infty, -x_{max})$, $\mathbf{R}_{\infty}= (x_{max}, \infty)$, $\mathbf{R}_{-1} = [-x_{max}, -x_{min}]$, $\mathbf{R}_{1} = [x_{min}, x_{max}]$, $\mathbf{R}_{\mathbf{0}}= (-x_{min}, x_{min})$
- Možemo da zapišemo:
- $\gamma x = -x_{max}$, $x \in \mathbf{R}\text{-}\infty$
- $\gamma x = x_{max}$, $x \in \mathbf{R}\infty$
- $\gamma x = 0$, $x \in \mathbf{R}0$
- Dakle, samo se podskupovi $\mathbf{R}_{-1}$ i $\mathbf{R}_{1}$ mogu prikazati u računaru, ali ne i tačno, u opštem slučaju. Naime, moguće je registrovati samo prvih m značajnih cifara broja, gde je m određeno dužinom mantise mašinskog broja. Već smo videli da za organizaciju smeštanja realnih brojeva, m je 23 u binarnom brojnom sistemu, što približno odgovara 7 u dekadnom.

Primer 8

- Proceniti interne vrednosti (mašinske brojeve) brojeva
- $x= 4.52\text{-}10^{41}$
- $y=-1.46\text{-}10^{-41}$
- $z=1/3$
- $\pi=3.14159265. . .$
- u dekadnom obliku, ako se realni brojevi smeštaju u četvorobajtne lokacije .

- Već smo za datu organizaciju predstavljanja realnih brojeva u računaru odredili:
- $xmax \approx 1.708\text{-}10^{38}$
- $xmin \approx 1.46\text{-}10^{-39}$
- broj značajnih cifara, m=7.
- Tako imamo,

$$x=4.52-10^{41} > x_{max} \Rightarrow x \in \mathbf{R}_{\infty} \Rightarrow \gamma x \approx 1.708-10^{38}$$

$$|y|=1.46-10^{-41} < x_{min} \Rightarrow \gamma y=0$$

$$z=0.333. . . , \quad \gamma z = 0.3333333$$

$$\gamma\pi = \begin{cases} 3.141592 & \text{(odsecanje)} \\ 3.141593 & \text{(zaokruzivanje)} \end{cases}$$

- Pri izvođenju niza računskih operacija u ALU, svaki delimičan rezultat se pamti u memorijskoj lokaciji kapaciteta m značajnih cifara, tj. sa tačnošću od m značajnih cifara.
- Očigledno je da, i ako polazni podaci mogu biti sasvim tačno predstavljani u računaru, *krajnji rezultat nekog proračuna ne mora biti tačan, zbog zaokruživanja svakog međurezultata na m značajnih cifara.*
- Tako u zaključku možemo da konstatujemo da ***u kompjuterskoj aritmetici ne važe zakoni asocijativnosti i distributivnosti.***