Brojni sistemi

Namenski računarski sistemi

Poziciona notacija

- Svaka pozicija broja ima određenu težinu
- Decimalni (dekadni) sistem brojeva
  - $2056 = 2 * 1000 + 0 * 100 + 5 * 10 + 6 * 1$
  - Može se predstaviti polinomom
  - $2056 = 2 * 10^3 + 0 * 10^2 + 5 * 10^1 + 6 * 10^0$
- Oktalni brojni sistem
  - $2056_8 = 2 * 8^3 + 0 * 8^2 + 5 * 8^1 + 6 * 8^0$
  - $2056_8 = 2 * 512 + 0 * 64 + 5 * 8 + 6 = 1070_{10}$
- Binarni brojni system
  - $011010_2 = 0* 2^5 + 1* 2^4 + 1* 2^3 + 0* 2^2 + 1* 2^1 + 0* 2^0$
  - $011010_2 = 0* 32 + 1*16 + 1* 8 + 0* 4 + 1* 2 + 0*1 = 26_{10}$
- Heksadekadni sistem
  - $2A3E = 2*16^3+10*16^2+3*16^1+14*16^0$
  - $2A3E = 2*4096+10*256+3*16+14*1$

Konvertovanje iz binarnog u dekadni brojni sistem

- Broj 011010 se može zapisati u obliku: $0*2^5+1*2^4+1*2^3+0*2^2+1*2+0$
- Ovaj izraz se izvlačenjem osnove ispred zagrade može zapisati i u obliku: $(0*2^4+1*2^3+1*2^2+0*2+1)*2+0$, tj. $((0*2^3+1*2^2+1*2+0)*2+1)*2+0$, tj. $(((0*2^2+1*2+1)*2+0)*2+1)*2+0$, tj. $((((0*2+1)*2+1)*2+0)*2+1)*2+0$
- Iz poslednjeg izraza vidi se algoritam:
  - Počinje se sa brojem 0 sa leve strane, koji predstavlja rezultat
  - Trenutni rezultat se množi sa 2 i dodaje sledeća cifra sa leve strane
  - Ovo se ponavlja sve dok se ne obradi i poslednja cifra

- **Primer:** 011010

  rez = 0

  rez = rez*2+ 1= 0*2+1=1

  rez = rez*2+ 1= 1*2+1=3

  rez = rez*2+ 0= 3*2+0=6

  rez = rez*2+ 1= 6*2+1=13

  rez = rez*2+ 0= 13*2+0=26
- Konačan rezultat je **26**

Konvertovanje iz dekadnog u binarni brojni sistem

- Broj 26, može se zapisati u binarnom obliku kao 011010
- Broj 011010 se može predstaviti kao (pogledati prosli slajd):
  $0*2^5+1*2^4+1*2^3+0*2^2+1*2+0$,
  $(0*2^4+1*2^3+1*2^2+0*2+1)*2+0$,
- Odavde se vidi da kada se 26 podeli sa 2, dobija se količnik 13 (13 = $0*2^4+1*2^3+1*2^2+0*2+1$)
- Ostatak pri deljenju sa 2 je predstavlja poslednju cifru broja (u ovom slucaju 0)
- Daljim deljenjem broja 13, dobijaju se ostale cifre rezultata
- Postupak se ponavlja dok ne ostane broj 0
- Dobijeni ostaci se čitaju od nazad

- Primer: broj 26
  26 = 13*2+0
  13 = 6*2+1
  6 = 3*2+0
  3 = 1*2+1
  1 = 0*2+1
  0 = kraj...
- Rezultat je 11010_2

Komplement dva predstava broja

- Koristi se za predstavljanje negativnih brojeva na računaru
- Jako je bitno da znamo koliko broj maksimalno zauzima cifara!!!
- Matematička osnova za binarni sistem sa 8 cifara:
  - Neka je dat broj X = 5, koji se binarno zapisuje kao 00000101
  - Za broj -5 važi da je -5 = 0-5 = $00000000_2 - 00000101_2$
  - Ako smo ograničeni na 8 cifara, tada se broj 0 može zapisati i kao 100000000_2, jer prvu cifru 1 nemamo gde da zapišemo, pa ostaje samo 00000000 ☺☺☺
  - Kako 100000000 ima 9 cifara, umesto njega koristimo zbir brojeva 100000000 = 11111111 + 00000001, jer oni imaju po 8 cifara
  - -5 = 0 – 5 = 00000000 – 00000101 = 11111111 + 00000001 – 00000101 = (11111111 – 00000101) + 00000001 = 11111010 + 00000001 = 11111011
  - Broj 11111010 naziva se nepotpuni (prvi) komplement broja -5, a broj 11111011 potpuni (drugi) komplement

Komplement dva predstava broja

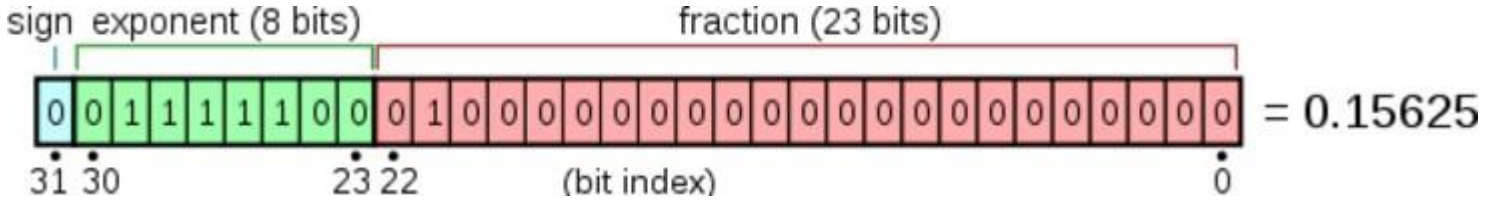
- Algoritam zapisa negativnog broja u binarnom sistemu
  - Absolutnu vrednost broja zapisati u binarnom sistemu sa maksimalnim brojem cifara (npr 8, 16, 32, ...)
  - Izračunati nepotpuni komplement inverovanjem svih njegovih cifara 0 -> 1, 1->0. Ovo je analogno oduzimanju datog broja od broja 1111...1
  - Izračunati potpuni komplement dodavanjem broja 1
- Primer: zapisati broj -26 u binarnom obliku, sa binarnih 8 cifara
  - abs(-26) = 26 = 00011010_2
  - Invertovanjem broja 00011010 dobija se 11100101
  - Dodavanjem broj 1 dobija se 11100101+00000001 = 11100110
  - Broj -26 zapisuje se kao 11100110
- Provera: 32-26 = 00100000_2 + 11100110_2 = 100000110_2 = 00000110_2= 6_{10}
  - Cifra 1 u broju 100000110_2 može da se briše, jer za nju nemamo dovoljno memorije gde da je smestimo (imamo sa 8 bitova)

XDR – External Data Representation standard

- XDR je standard koji je doneo IETF. Aktuelna verzija je RFC 4506
- XDR se koristi za serializaciju različitih tipova podataka podataka (integer, boolean, string, float...) , radi razmene između računara različitih arhitektura.
- Kao osnovnu jedinicu prenosa XDR koristi 4-bajta, koji su serijalizovani u Big-endian redosledu.
- Podaci manji od 4 bajta, nakon konverzije zauzimaju svaki po 4 bajta.
- Za podatke čija je dužina proizvoljna (npr. string), nakon konverzije je ukupna dužina podataka deljiva sa 4, tj. podaci se na kraju po potrebi dopunjavaju praznim bajtima .

IEEE 754 – standard za aritmetiku u pokretnom zarezu 1/3

- Format – definiše pretstavu brojeva u pokretnom zarezu , koja se koristi da pretstavi podskup realnih brojeva. Format je opisan svojom bazom, preciznošću i opsegom eksponenta. Uglavnom se koristi pomereni eksponent za neku vrednost (bias), tako da eksponent bude pozitivan broj
- Standard IEEE 754 definiše formate za decimalnu i binarnu bazu. Nama je akcenat na binarnoj bazi.
- Na sledećoj slici je prikazan jedan format veličine 32 bita (float)



- Zapis se tumači na sledeći način:

$$(-1)^{b_{31}} \times (1.b_{22}b_{21}...b_0)_2 \times 2^{b_{30}b_{29}...b_{23}-127}.$$

- Mantisa je normalizovana, tj. vrednost joj je između 1 i 2 (1 <= M < 2), a vodeća jedinica se ne čuva! Eksponent je ovde uvek uvećan za bias = 127

IEEE 754 – standard za aritmetiku u pokretnom zarezu 2/3

- Decimalna vrednost se dobija na sledeći način:

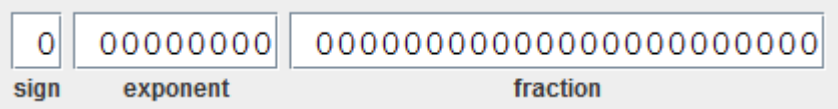
$$\text{value} = (-1)^{\text{sign}} \times \left(1 + \sum_{i=1}^{23} b_{23-i} 2^{-i}\right) \times 2^{(e-127)}.$$

- Standard IEEE 754 – Binarni formati za razmenu brojeva
  - koriste se za razmenu realnih brojeva između različitih sistema - različitih implementacija pokretnog zareza
  - XDR standard nalaže da se za razmenu brojeva u pokretnom zarezu koristi upravo IEEE 754 – binarni format za razmenu brojeva.
  - Definisana su tri formata za razmenu: binary32, binary64 i binary128

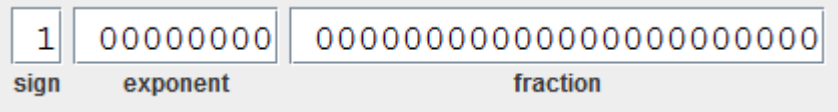
Binarni format	Znak	Eksponent	Eksponent bias	Mantisa
Binary32	1-bit	8- bit	127	23-bit
Binary64	1-bit	11-bit	1023	52-bit
binary128	1-bit	15-bit	16383	112-bit

IEEE 754 – standard za aritmetiku u pokretnom zarezu 3/3

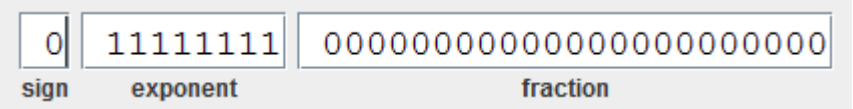
- Postoje specijalne vrednosti eksponenta koje kodiraju:
  - +0 – svi biti eksponenta su 0, bit znaka je 0, mantisa je 0



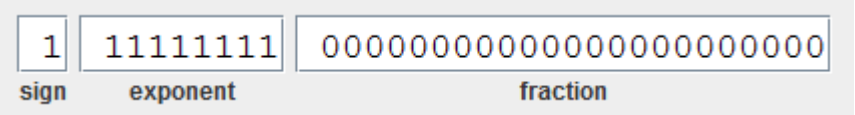
  - -0 – svi biti eksponenta su 0 i bit znaka je 1, mantisa je 0



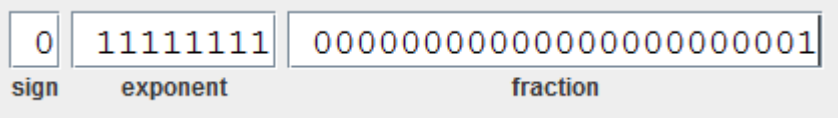
  - +∞ - svi biti eksponenta su 1,bit znaka je 0, mantisa je0



  - -∞ - svi biti eksponenta su 1,bit znaka je 1, mantisa je 0



  - NaN - svi biti eksponenta su 1,bit znaka je 0 ili 1, mantisa je različita od 0



Aritmetika u pokretnom zarezu u C programskom jeziku 1/2

- Tip float – single precision (jednostruka preciznost)
  - Najčešće se implementira kao IEEE 745 binary32, ali ne mora da bude. Kao i za druge tipove podataka standard nalaže minimalni podskup realnih brojeva.
- Tip double – double precision (dupla preciznost)
  - Najčešće kompajleri implementiraju kao IEEE 745 binary64, ali ne mora da bude. Kao i za druge tipove podataka standard nalaže minimalni podskup realnih brojeva.
- Tip long double – extended precision(proširena preciznost)
  - Najčešće kompajleri implementiraju kao 80-bitni format pomičnog zareza, koji nije po IEEE 745 standardu. Neki kompajleri ovaj tip tretiraju kao tip double. IEEE 745 binary128 se takođe koristi za neke procesore(npr. SPARC)
  - 80-bitni format je uveden u intel 8087 matematičkom koprocesoru i nije mu namena bila da u memoriji čuva realne brojeve u ovom formatu već da ovaj format koristi interno u samom koprocesoru da bi se umanjio gubitak preciznosti u toku matematičkih operaija.

Aritmetika u pokretnom zarezu u C programskom jeziku 2/2

- U sledećoj tabeli su predstavljeni najčešće korišćeni tipovi u C kompajlerima:

C tip	Naziv formata	Izgled formata	bias
Float	binary32	S EEEEEEEE MMMMMMMMMMMMMMMMMMMMMMM 0 1 8 9 31 sign bit, 8-bit exponent, 23-bit mantissa	127
Double	binary64	sign bit, 11-bit exponent, 52-bit mantissa	1023
Long double	80-bit float	sign bit, 15-bit exponent, 64-bit mantissa	16383

Zadaci za vežbanje

- **Zadatak 1:** Napisti program koji učitava broj X tipa int i konvertuje ga i binarni sistem u obliku stringa. Program treba da ima funkciju sa zaglavljem:

  char* konverzija(int x);

  - Funkciju realizovati na dva načina:
    - Koristeći matematičke operacije / i % da bi se dobio traženi broj
    - Pristupajući određenim bitovima početnog broja i čitajući ih koristeći I operaciju (AND, operator &).

- **Zadatak 2:** Napisati program koji učitava broj X u binarnom sistemu u obliku stringa i konvertuje taj broj u tip podataka int. Program treba da ima funkciju sa zaglavljem:

  int konverzija(char* s);

  - Funkciju realizovati na dva načina:
    - Koristeći matematičke operacije + i * da bi se dobio traženi broj
    - Pristupajući određenim bitovima rezultata i postavljajući ih koristeći ILI operaciju (OR, operator I).