




# Osnovi računarstva I

## Binarna aritmetika

Oduzimanje u binarnom brojnom sistemu

- 
- Oduzimanje se svodi na sabiranje:  $X - Y = X + (-Y)$
  - Sabiranje se jednostavno realizuje korišćenjem osnovnih logičkih kola.
  - Posmatrajmo analogni slučaj dekadnog oduzimanja kod trocifrenih brojeva:  $356 - 174$

- Oduzimajući na uobičajen način dobijamo:

$$356 - 174 = 356 + (-174) = 182$$

- Oduzimanje dva trocifrena broja može dati najviše trocifreni rezultat.
- Dodavanjem broja koji ima tri najniže nule nećemo promijeniti vrijednost značajnih cifara.
- Na osnovu toga gornji izraz možemo zapisati kao:

$$1000 + 356 - 174 = 356 + (1000 - 174)$$

- Razlika  $1000 - 174$  se (u ovom kontekstu) naziva komplement desetke dekadnog broja 174.
- Komplement desetke se, po definiciji, dobija dodavanjem jedinice na komplement devetke istog dekadnog broja.

- Dakle, komplement devetke trocifrenog dekadnog broja definiše se razlikom maksimalnog trocifrenog dekadnog broja (999) i broja čiji komplement devetke tražimo.
- Drugim riječima, razlika  $999 - 174$  predstavlja komplement devetke dekadnog broja 174, dok je:  


$$\text{komplement desetke (174)} = \text{komplement devetke (174)} + 1 = 999 - 174 + 1 = 1000 - 174.$$
- Izračunavanjem komplementa devetke, svaka pojedinačna cifra dekadnog broja oduzima se od maksimalne dekadne cifre 9 (komplementira se). Time se problem svodi na oduzimanje jednocifrenih brojeva, odnosno na dopunu svake pojedinačne cifre početnog dekadnog broja do 9. Otuda i potiče naziv "komplement devetke".
- Izračunavanjem poslednjeg izraza dobija se:

$$\begin{array}{r} 999 + 1 \\ - 174 \\ \hline = 825 + 1 = 826 \end{array}$$

- 
- Drugim riječima, tražena razlika dekadnih brojeva iz našeg primjera je:

$$356 - 174 = 356 + (1000 - 174) = 356 + 826 = 1182.$$

- Rezultat oduzimanja trocifrenih cijelih brojeva ne može biti četvorocifren broj, pa je jasno da dobijeni rezultat treba dodatno tumačiti.
- Vrijednost 1000 koju smo dodali u procesu izračunavanja sada treba "eliminirati". U tom cilju ćemo zanemariti vodeću jedinicu, pa se dobija rezultat 182, što je upravo traženi rezultat.
- Postojanje četvrte cifre (jedinice) znači da je rezultat **pozitivan!!!**
- Ako nema četvrte jedinice (rezultat je sa tri ili manje značajnih cifara) rezultat je **negativan**: potrebno je naći njegov komplement desetke i staviti znak minus.
- U opštem slučaju:
  - Ako oduzimamo dva N-tocifrena broja i dobijemo rezultat sa N+1 značajnom cifrom znači da je rezultat pozitivan i najviša cifra se odbacuje.
  - U suprotnom, traži se komplement desetke i dodaje se znak minus.

- 
- *Primjer*: Korišćenjem metoda sa komplementom desetke izračunati razliku dekadnih brojeva 24–53.

- *Rješenje*:

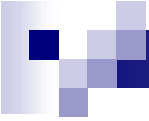
- Komplement devetke umanjioaca je  $99-53=46$ .
- Komplement desetke istog broja je  $46+1=47$ .
- Sabiranjem komplementa desetke umanjioaca sa umanjenikom dobijamo:

$$24+47=71.$$

- Rezultat 71 je dvocifren broj, odnosno ima isti broj cifara kao umanjenik i umanjilac. To znači da je rezultat oduzimanja negativan, pa treba pronaći komplement desetke ovog rezultata:

$$99-71+1=29$$

- Rezultat je sa znakom minus:  $(-29)$ .

- 
- Analogni postupak se primjenjuje kod binarnih brojeva, samo što je postupak jednostavniji (postoje samo dvije cifre – 0 i 1).
  - Umjesto komplementa devetke → jedinični komplement.
  - Umjesto komplementa desetke → dvojni komplement.
  - *Primjer*: Naći jedinični komplement binarnog broja 101101.
  - *Rješenje*: komplement jedinice (često nazivan jediničnim komplementom) proizvoljnog binarnog broja dobija se oduzimanjem ovog broja od maksimalnog binarnog broja, koji ima isti broj cifara kao broj čiji se komplement traži. Maksimalni binarni broj sa određenim brojem cifara je onaj čije su sve cifre jednake jedinici.

$$\begin{array}{r} 111111 \\ -101101 \\ \hline 010010 \end{array}$$

- Dvojni komplement cijelog binarnog broja dobija se sabiranjem jedinice sa jediničnim komplementom toga broja. Tako je dvojni komplement broja 101101 jednak:  $010010 + 1 = 010011$ .
- **Primjer:** Izračunati  $10111_{(2)} - 10001_{(2)}$  računajući u binarnom brojnom sistemu.
- Rješenje:

$$10111 - 10001 = 10111 + (-10001)$$

Ako se doda šestocifreni broj 100000 i zanemari najviša cifra (jedinica), rezultat se ne bi promijenio:

$$10111 - 10001 \rightarrow 10111 + (-10001 + 100000)$$

↓  
Dvojni komplement

Znači, umanjenik je potrebno sabrati sa dvojnim komplementom umanjioaca.

- 
- Dvojni komplement broja 10001 je:

$$11111 - 10001 + 1 = 01110 + 1 = 01111$$

- Sabiranjem dvojnog komplementa umanjioaca sa umanjenikom dobijamo rezultat tražene operacije:

$$\begin{array}{r} 10111 \\ + 01111 \\ \hline \cancel{1}00110 \end{array}$$

- Dobijeni rezultat je  $00110_{(2)}$ , a odbačena jedinica na krajnjem lijevom mjestu (tzv. pretek – prenos sa mjesta najveće težine kod posmatranih petobitnih brojeva) označava samo da je rezultat operacije oduzimanja pozitivan.



- 
- *Primjer*: Izvršiti binarno oduzimanje  $10-101$ .

- Rješenje:

- Jedinični komplement umanjioaca  $101$  je  $010$ , a njegov dvojni komplement je:  $010 + 1 = 011$ .

- Sabiranjem dvojnog komplementa umanjioaca sa umanjenikom dobijamo:

$$\begin{array}{r} 10 \\ + 011 \\ \hline 101 \end{array}$$

- Pošto nije bilo preteka dobijeni rezultat je negativan i potrebno ga je dalje tumačiti – izračunavanjem dvojnog komplementa dobijenog rezultata i, potom, dodavanjem predznaka minus ispred njega.

- Rezultat:

$$010 + 1 = 011_{(2)} \rightarrow -3_{(10)}$$

- *Primjer*: Izvršiti oduzimanje binarnih brojeva koji uz cjelobrojni dio broja sadrže i decimalni dio 10110.01–1011.1
- Rješenje:
  - Prvo treba **izjednačiti** broj binarnih cifara
  - Jedinični komplement umanjioaca pronalazi se komplementiranjem svakog bita, dok se dvojni komplement istog broja dobija dodavanjem jedinice na poziciji najmanje težine pronađenog jediničnog komplementa umanjioaca:

$$\begin{array}{r}
 \underline{01011.10} \quad \text{umanjilac} \\
 10100.01 \quad \text{jedinični komplement umanjioaca} \\
 + \quad 0.01 \\
 \hline
 10100.10 \quad \text{dvojni komplement umanjioaca.}
 \end{array}$$

- 
- Sabrati umanjenik sa dvojnim komplementom umanjioca:

$$\begin{array}{r} 10110.01 \\ + 10100.10 \\ \hline 1\ 01010.11 \end{array}$$

- Pošto u rezultatu postoji pretek, rezultat je pozitivan i iznosi (nakon odbacivanja preteka):


$$01010.11_{(2)} = 10.75_{(10)}$$



# Osnovi računarstva I

**Binarna aritmetika**

Množenje u binarnom brojnom sistemu

- 
- Množenje u binarnom brojnom sistemu obavlja se na isti način kao i u dekadnom brojnom sistemu.
  - *Primjer*: Izračunati  $1011_{(2)} \times 101_{(2)}$  računajući u binarnom brojnom sistemu.
  - Rješenje:

$$\begin{array}{r} 1011 \times 101 = \quad 1011 \\ \phantom{1011 \times 101 =} \quad 0000 \\ \phantom{1011 \times 101 =} \quad 1011 \\ \hline \phantom{1011 \times 101 =} 110111 \end{array}$$

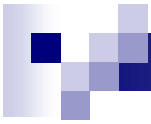
- Množenje se u binarnom brojnom sistemu može realizovati sabiranjem i pomjeranjem jednog od operanada i to onoliko puta koliko drugi operand ima cifara.



# Osnovi računarstva I

## Binarna aritmetika

Dijeljenje u binarnom brojnom sistemu



- Postupak dijeljenja u binarnom brojnom sistemu veoma je sličan postupku dijeljenja višecifrenih dekadnih brojeva
- *Primjer*: Izvršiti operaciju dijeljenja dekadnih brojeva 14 i 4 u binarnom brojnom sistemu.
- Rješenje:

$$\begin{array}{r} 1110 : 100 = 11.1 \\ \underline{- 100} \\ 110 \\ \underline{- 100} \\ 100 \\ \underline{- 100} \\ 000 \end{array}$$



# Osnovi računarstva I

Formati binarnog zapisivanja  
podataka



# Format podataka

- Jedna binarna cifra (cifra 1, odnosno cifra 0) naziva se **bit** (*binary digit*).
- Na primjer, za broj 10110 kažemo da je petobitni, dok, u opštem slučaju, za broj sa  $N$  bitova kažemo da je  $N$ -tobitni broj.
- Prva cifra s lijeve strane ima najveću težinu i naziva se najznačajnijim bitom (*Most Significant Bit* - MSB).
- Prva cifra s desne strane ima najmanju težinu i naziva se najmanje značajanim bitom (*Least Significant Bit* - LSB).
- Skup bitova koji čini jednu cjelinu naziva se riječ (*word*). Pošto se riječi uobičajeno čuvaju u memoriji, često se koristi termin "memorijska riječ".
- Tipičan broj bitova od kojih se sastoji jedna riječ je oblika  $2^n$ . Uobičajene vrijednosti su 8, 16, 32, 64..., bita.
- Skup od osam bitova naziva se **bajt** (*byte*).



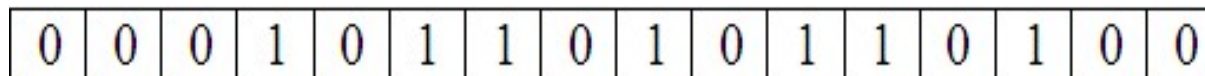
8 bitova → bajt

# Format podataka

- Bajt se, po pravilu, koristi kao mjerna jedinica bez obzira na dužinu riječi. Tako se, na primjer, riječ od 16 bitova posmatra kao riječ sastavljena od 2 bajta:



- Sadržaj memorijske riječi može se tumačiti na razne načine. Jedan način je tumačenje sadržaja memorijske riječi kao binarnog broja. Tako bi memorijska riječ:



mogla da se protumači kao broj čija je vrijednost:

$$2^{12} + 2^{10} + 2^9 + 2^7 + 2^5 + 2^4 + 2^2$$



# Format podataka

- Međutim, pored prethodnog tumačenja, svaka kombinacija jedinica i nula, u okviru raspoloživih  $N$  bitova, može biti protumačena od strane računara i kao određeno slovo, matematički ili interpukcijski znak, ili pak shvaćena kao naredba da računar nešto uradi.



# Zapis brojeva

- Neoznačeni cijeli brojevi (cijeli pozitivni brojevi, odnosno prirodni brojevi uključujući nulu).
- Označeni cijeli brojevi (cijeli brojevi sa znakom).
- Decimalni brojevi.



# Zapis neoznačenih cijelih brojeva

- Memorijska riječ veličine  $N$  bitova.
- Minimalni neoznačeni cijeli broj koji se može zapisati u ovoj riječi sastoji se od  $N$  binarnih nula i ima dekadnu vrijednost  $0_{(10)}$ .
- Maksimalni neoznačeni cijeli broj koji se može smjestiti u ovoj riječi sastoji se od  $N$  binarnih jedinica i ima dekadnu vrijednost  $(2^N - 1)_{(10)}$ .
- Drugim riječima, u  $N$ -tobitnoj memorijskoj riječi mogu se zapisati neoznačeni cijeli brojevi iz dekadnog opsega  $0_{(10)}$  do  $(2^N - 1)_{(10)}$ .

# Zapis neoznačenih cijelih brojeva

- Na primjer, u 16-tobitnoj riječi je moguće smjestiti ove brojeve:

$$00000000 \ 00000000 = 0000_{(16)} = 0_{(10)}$$

$$00000000 \ 00000001 = 0001_{(16)} = 1_{(10)}$$

$$00000000 \ 00000010 = 0002_{(16)} = 2_{(10)}$$

...

...

...

$$11111111 \ 11111111 = \text{FFFF}_{(16)} = (2^{16} - 1)_{(10)} = 65535_{(10)}$$



# Zapis označenih cijelih brojeva

- Potrebno je, na neki način, omogućiti zapisivanje znaka broja.
- Postoji više načina na koji se mogu zapisati brojevi sa znakom.
- Danas je gotovo isključivo u upotrebi tzv. *zapis u dvojnomo komplementu*. Kod ovog zapisa najznačajniji bit (*MSB*) rezerviše se za zapisivanje znaka broja (u literaturi nazvan *sign bit*, odnosno *bit znaka*).
  - Ukoliko *MSB* ima vrijednost 0, broj se tretira kao pozitivan i *predstavljen je u svom originalnom obliku*.
  - Ukoliko *MSB* ima vrijednost 1 broj je negativan i *predstavljen je u svom dvojnomo komplementu*.

## Zapis **pozitivnih** cijelih brojeva

- *MSB* ima vrijednost nule.
- Preostalih  $(N - 1)$  bitova koriste se za zapisivanje apsolutne vrijednosti broja iz opsega od  $0_{(10)}$  do  $(2^{N-1}-1)_{(10)}$ .


## Zapis **negativnih** cijelih brojeva

- *MSB* ima vrijednost jedinice.
- Maksimalni negativni cijeli broj sastoji od  $N$  jedinica i ima dekadnu vrijednost  $(-1)_{(10)}$
- Minimalni (negativni) cijeli ima jedinicu na mjestu *MSB*, a na preostalim mjestima nule:  $(-2^{N-1})_{(10)}$

***Dozvoljeni opseg  $N$ -tobitne memorijske riječi za zapis označenih cijelih brojeva:***

***$(-2^{N-1})_{(10)}$  do  $(2^{N-1}-1)_{(10)}$***





U 16-bitnoj memorijskoj riječi, mogu se smjestiti sljedeći označeni cijeli brojevi:

Pozitivni: 00000000 00000000 =  $0_{(10)}$

00000000 00000001 =  $1_{(10)}$

00000000 00000010 =  $2_{(10)}$

... ..

01111111 11111111 =  $(2^{15} - 1)_{(10)} = 32767_{(10)}$

Negativni: 11111111 11111111 =  $-1_{(10)}$

11111111 11111110 =  $-2_{(10)}$

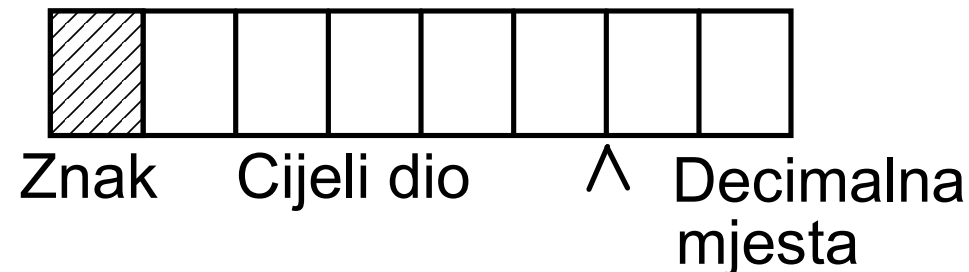
... ..

10000000 00000001 =  $(-2^{15} - 1)_{(10)}$

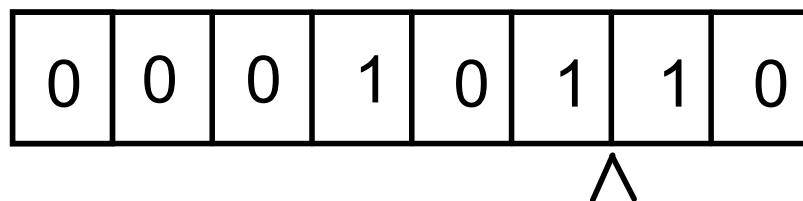
10000000 00000000 =  $(-2^{15})_{(10)} = -32768_{(10)}$

# Zapis decimalnih brojeva sa nepomičnim zarezom

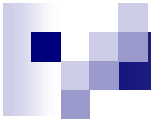
- Pozicija decimalnog zareza podrazumijeva se na fiksnom (tačno određenom) mjestu.
- Brojevi se tumače u skladu sa usvojenom pozicijom decimalnog zareza.
- Na primjer, ako se podrazumijeva da 8-bitni zapis u računaru ima dva decimalna mjesta, onda struktura zapisa označenog decimalnog broja izgleda:



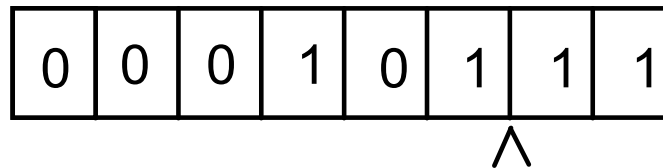
- *Primjer*: Binarni broj  $x=+101.1011$  može se, u opisanom formatu, zapisati na sljedeći način:



- Zapisani broj označava se sa  $Q[x]$  (tzv. „kvantizirano  $x$ “).
- On se razlikuje od originalnog broja  $x$ , zato što su za zapisivanje decimalnog dijela broja predviđena dva mjesta, a broj  $x$  ima četiri decimale. Izvršeno je tzv. *odsijecanje*.
- Greška napravljena prilikom zapisivanja (tzv. **greška kvantizacije**) iznosi:  $\varepsilon = x - Q[x]$ .
- U gornjem primjeru, greška kvantizacije iznosi  $\varepsilon = 0.0011_{(2)}$



- Prilikom zapisivanja binarnog broja može se primijeniti i metod **zaokruživanja** na mjestu najmanje značajnog bita.
- *LSB* uzima vrijednost jedinice ukoliko je prva cifra odsječenog dijela broja jedinica, odnosno vrijednost nule ukoliko je prva cifra odsječenog dijela broja 0:



- U gornjem primjeru, greška kvantizacije iznosi  $\varepsilon = -0.0001_{(2)}$



# Zapis decimalnih brojeva sa pomičnim zarezom

- Veliki dekadni brojevi obično se zapisuju pomoću eksponencijalnog zapisa (tzv. naučna notacija).
- Na primjer, dekadni broj 120000000000 može se zapisati kao
$$1.2 \times 10^{11}$$
- Dekadni broj 0.0000000378 zapisujemo kao  $3.78 \times 10^{-8}$
- Vodeći, decimalni, dio broja (u prvom slučaju 1.2, a u drugom 3.78), predstavljen u formatu zapisivanja sa nepomičnim zarezom, naziva se *mantisa*.
- Eksponent desetke, predstavljen označenim cijelim brojem koji odgovara stvarnom položaju decimalnog zareza, naziva se *eksponent*.

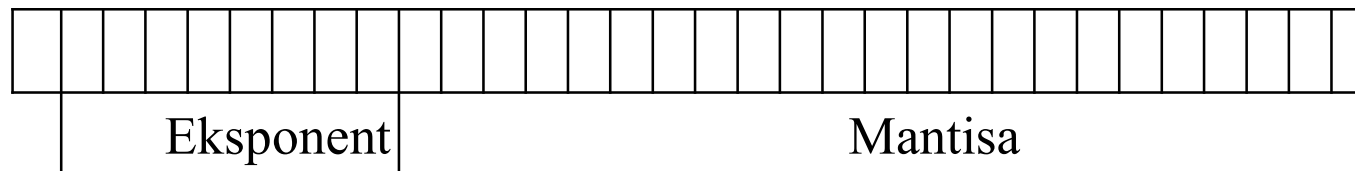
# Zapis decimalnih brojeva sa pomičnim zarezom

- Slična logika primjenjuje se kod binarnih brojeva:

$$1101000000 = 1.101 \times 2^{10}$$

$$0.000000001001 = 1.001 \times 2^{-9}$$

- Jedna od mogućih struktura zapisa binarnog broja sa pomičnim zarezom prikazana je na sljedećoj slici:





# Zapis decimalnih brojeva sa pomičnim zarezom

- IEEE 754 standard:
  - mantisa 23 bita
  - eksponent 8 bita
  - 1 bit se koristi za zapisivanje znaka broja (ako je bit znaka jednak nuli → pozitivan broj, a ako je bit znaka jednak jedinici → negativni broj)
  - Mantisa se zapisuje u tzv. normalizovanom obliku
  - Eksponent se ponderiše da bi se prikazivali i pozitivni i negativni brojevi.



# Zapis brojeva u BCD kodu

- BCD (*Binary Coded Decimal*)
- Svaka dekadna cifra pojedinačno se zapisuje skupom od po 4 binarne cifre:

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





# Zapis brojeva u BCD kodu

- Ostale kombinacije od po četiri bita (1010, 1011, 1100, 1101, 1110 i 1111) nijesu dozvoljene, odnosno nemaju smisla u BCD kodu.
- *Primjer*: Zapisati brojeve  $2873_{(10)}$  i  $5745_{(10)}$  u BCD kodu, a zatim pronaći njihov zbir.
- Rješenje:

$$2873_{(10)} \rightarrow 0010 \ 1000 \ 0111 \ 0011$$

$$5745_{(10)} \rightarrow 0101 \ 0111 \ 0100 \ 0101$$

## Sabiranje brojeva u BCD kodu

- Sabiranjem reprezentovanih binarnih zapisa dobija se:

$$\begin{array}{r} 0010\ 1000\ 0111\ 0011 \\ +\ 0101\ 0111\ 0100\ 0101 \\ \hline 0111\ 1111\ 1011\ 1000 \\ 7\ \text{nedo.} \quad \text{nedo.} \quad 8 \end{array}$$

- Posmatrajmo sve moguće nedozvoljene binarne tetrade, njihov dekadni i željeni zapis u BCD kodu



## Sabiranje brojeva u BCD kodu

1010 odgovara broju **10**, odnosno u BCD kodu 0001 0000,  
1011 odgovara broju **11**, odnosno u BCD kodu 0001 0001,  
1100 odgovara broju **12**, odnosno u BCD kodu 0001 0010,  
1101 odgovara broju **13**, odnosno u BCD kodu 0001 0011,  
1110 odgovara broju **14**, odnosno u BCD kodu 0001 0100,  
1111 odgovara broju **15**, odnosno u BCD kodu 0001 0101.

U svakom od ovih slučajeva potrebno je, u cilju ispravnog tumačenja rezultata, nedozvoljenoj tetradi dodati  $6_{(10)} = 0110_{(2)}$  da bi se dobio korektan rezultat i prenos na sljedeću tetradu.

Na primjer:

$$1010 + 0110 = 1\ 0000,$$
$$1011 + 0110 = 1\ 0001,$$
$$1100 + 0110 = 1\ 0010, \dots$$

## Sabiranje brojeva u BCD kodu

- Prema tome, tačan rezultat u prethodnom primjeru dobija se dodavanjem tetrade  $0110_{(2)}$  na mjestima gdje se dobijaju nedozvoljene tetrade u BCD zapisu:

$$\begin{array}{r} 0010\ 1000\ 0111\ 0011 \\ +\ 0101\ 0111\ 0100\ 0101 \\ \hline 0111\ 1111\ 1011\ 1000 \\ +\ \quad\quad 0110\ 0110 \\ \hline 1000\ 0110\ 0001\ 1000 \\ 8\quad\quad 6\quad\quad 1\quad\quad 8. \end{array}$$

## Sabiranje brojeva u BCD kodu

- Sabiranjem dekadnih cifara može se dobiti i rezultat koji se ne može predstaviti ni kada se tetrađa tumači u binarnom obliku (vrijednost između 16 (npr. sabiranjem cifara 9 i 7) i 18 (sabiranjem cifara 9 i 9), koje ne mogu biti predstavljene binarnom tetradom).

broju **16** odgovara binarno 1 0000, tj. BCD 0001 0110,

broju **17** odgovara binarno 1 0001, tj. BCD 0001 0111,

broju **18** odgovara binarno 1 0010, tj. BCD 0001 1000.

u svim gornjim slučajevima postoji prenos na višu binarnu tetradu, te je u cilju dobijanja ispravnog rezultata u BCD kodu, na nižu binarnu tetradu dobijenog rezultata neophodno dodati

$6_{(10)}$ , odnosno  $0110_{(2)}$  :



## Sabiranje brojeva u BCD kodu

$$1\ 0000 + 0110 = 0001\ 0110$$

$$1\ 0001 + 0110 = 0001\ 0111$$

$$1\ 0010 + 0110 = 0001\ 1000.$$

- Uopšteno, u praksi se ovaj postupak svodi na sljedeće:
  - Uoči se da li postoji prenos sa tekuće na višu tetradu u prvom nivou sabiranja (prije dodavanja 0110 na nedozvoljenim stanjima) i
  - Ukoliko postoji prenos, na tekuću tetradu (sa koje je došlo do prenosa) se dodaje  $0110_{(2)}$



## Zapis karaktera - kodovi

- Računar radi ne samo sa brojevima, već i sa tekstom, odnosno sa alfanumeričkim podacima.
- Određena kombinacija jedinica i nula ne mora predstavljati samo binarni broj, već može predstavljati karakter (alfanumerički podatak – slovo, cifru, znak interpunkcije, ...).
- Sa  $N$  bita u memorijskoj riječi moguće je predstaviti  $2^N$  različitih karaktera.
- Prvi standardizovani i široko prihvaćen kod za predstavljanje karaktera u računarima i računarskoj opremi je bio **ASCII** kod (*American Standard Code for Information Interchange*).

# Zapis karaktera - kodovi

- Dio ASCII tabele:

1000001	65	41	A
1000010	66	42	B
1000011	67	43	C
1000100	68	44	D
1000101	69	45	E
1000110	70	46	F
1000111	71	47	G
1001000	72	48	H
1001001	73	49	I
1001010	74	4A	J
1001011	75	4B	K
1001100	76	4C	L
1001101	77	4D	M