Programiranje 1

Predavanje 03 - Prikaz podataka u računalu, cijeli brojevi bez predznaka, cijeli brojevi s predznakom

Matej Mihelčić

Prirodoslovno-matematički fakultet
Matematički odsjek

9. listopada 2023.

Osnovni tipovi podataka u računalu

Osnovni tipovi podataka u računalu su **cjeline (blokovi)** bitova s kojima računalo može (zna) raditi neovisno o njihovom sadržaju.

Primjer: osnovne cjeline koje možemo adresirati.

Postoje instrukcije koje rade s tim cjelinama bez obzira na njihov **tip** (interpretaciju, značenje sadržaja).

Primjer: instrukcije za transfer tih cjelina između memorije i registara procesora.

Računalo može raditi i s većim i s manjim cjelinama (ovisno o veličini stranice, **byte se ne dijeli**).

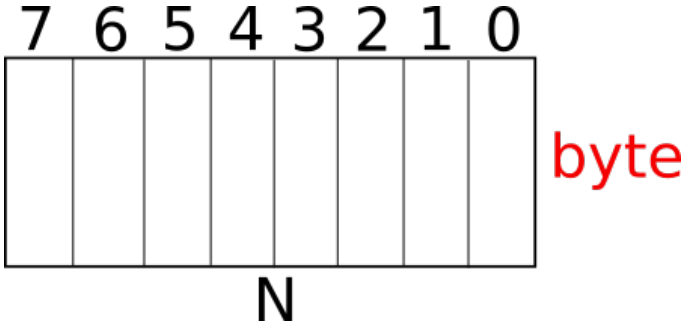
Osnovni tipovi **ovise o arhitekturi računala**. Jedina informacija koju imamo o osnovnim tipovima je njihova duljina u bitovima.

Osnovni tipovi podataka u računalu

Primjer: Intelova 32-bitna arhitektura $IA - 32$.

Osnovna cjelina (duljina stranice): 1 byte = 8 bitova.

Naziv	Duljina (bita)	Broj byteova
word	16	2
doubleword	32	4
quadword	64	8
double quadword	128	16



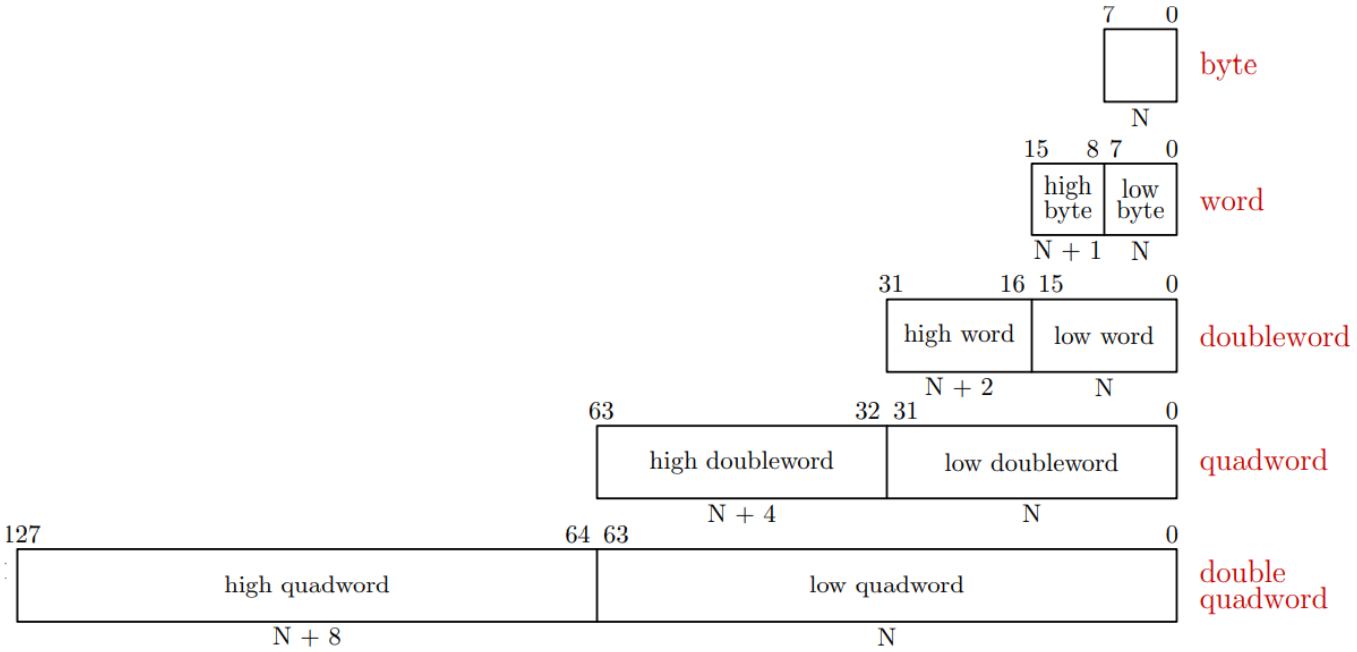
oznake za bitove - standardno.
način adresiranja - specifično za pojedinu arhitekturu.

Osnovni tipovi podataka u računalu

- Najniži byteovi na najnižim adresama - *little endian* encoding.
- Vodeći dio na najnižim adresama - *big endian* encoding.

Standardno u *C*-u: **najniža adresa** (na kojoj počinje cjelina) je ujedno i **adresa čitave cjeline**.

Osnovni tipovi podataka IA-32:



Jednostavni tipovi podataka

Za računanje trebamo: a) dodatnu interpretaciju sadržaja (cjelina, bitova), b) operacije s takvom vrstom podataka.

Skup podataka i operacije nad njima čine neku algebarsku strukturu (**tip podatka**).

Jednostavni tipovi podataka: tipovi podataka za koje računalo može **prikazati pripadni skup podataka** i **izvesti pripadne operacije nad njima**.

Podjela: a) nenumerički tipovi (znakovni, logički), b) numerički (cjelobrojni, realni).
Nenumerički tipovi se svode na numeričke.

Nenumerički tipovi podataka

Znakovi:

- Prikaz u nekom **kodu** (obično nadskup ASCII koda) - **cijeli brojevi**.
- Nema operacija sa znakovima (osim prebacivanja u memoriji). **Operacije se svode na elementarne operacije nad cijelim brojevima**.

Znakovima dobivamo čovjeku razumljiv ulaz/izlaz, koriste se i kao **dijelovi složenijih struktura podataka**.

Primjer korištenja znakova u C-u

```
1   #include <stdio.h>
2
3   int main(void){
4         char c = '2';
5
6         printf("%c\n",c); /*2*/
7         printf("%d\n",c); /*50*/
8
9     return 0;
10  }
```

%*c* - piše vrijednost kao znak (`char`).
%*d* - piše vrijednost kao cijeli broj (`int`), decimalno.

Logički tip:

- Logičke vrijednosti prikazuju se bitovima: (laž - 0, istina - 1). Možemo prikazati i kao cijele brojeve.
- Osnovne operacije `ne` (NOT), `i` (AND), `ili` (OR) se svode na aritmetičke operacije u bazi 2.

Logičke operacije se mogu izvesti `bit-po-bit` nad skupinama bitova u većoj cjelini.

Mi ćemo koristiti logički tip i pripadne operacije za formulaciju i kombiniranje uvjeta u uvjetnim naredbama i petljama.

`C` nema posebni tip za logičke vrijednosti već izravno koristi cijele brojeve (laž - 0, istina - 1).

Primjer logičkih vrijednosti u C-u

```
1   #include <stdio.h>
2
3   int main(void){
4         int i=5, j=10;
5
6         printf("%d\n",i<j); /*1*/
7         printf("%d\n",i>=j); /*0*/
8         printf("%d\n",i==j); /*0*/
9         printf("%d\n",i&&j); /*1*/
10        printf("%d\n",i||j); /*1*/
11        printf("%d\n",!i); /*0*/
12        printf("%d\n",!j); /*0*/
13
14    return 0;
15  }
```

Numerički tipovi podataka

- Moraju realizirati četiri osnovne aritmetičke operacije $(+, -, \cdot, /)$ na raznim skupovima brojeva ($\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$).
- **Problem:** navedeni skupovi su beskonačni i **ne mogu se prikazati u računalu**.
- Možemo prikazati samo **konačne podskupove**, model beskonačnog skupa.

Tri glavne kategorije:

- *cijeli* brojevi bez predznaka - model za $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$.
- *cijeli* brojevi s predznakom - model za $\mathbb{Z}$.
- *realni* brojevi - model za $\mathbb{R}$.

Svaka grupa ima nekoliko **podtipova** ovisno o **veličini pripadnog konačnog skupa prikazivih brojeva** (ovisi o broju bitova predviđenih za prikaz).

Numerički tipovi podataka

Prijelaz na konačne skupove **mijenja realizaciju aritmetike na odgovarajućem skupu**.

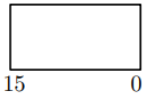
Za potpuni opis numeričkog tipa treba opisati:

- Koji konačni skupovi modeliraju odgovarajuće matematičke skupove.
- Kako se točno prikazuju njihovi elementi u računalu.
- Kako se realizira aritmetika na tim skupovima.

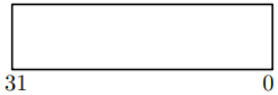
Primjer - numerički tipovi na IA-32

Cijeli broj bez predznaka (unsigned integer)

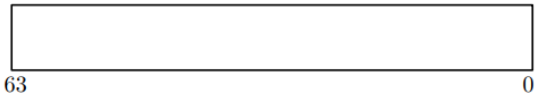
byte unsigned integer



word unsigned integer

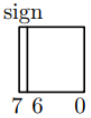


doubleword unsigned integer

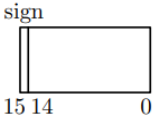


quadword unsigned integer

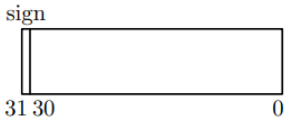
Cijeli broj s predznakom (signed integer)



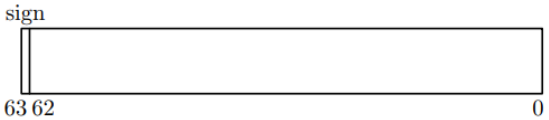
byte signed integer



word signed integer

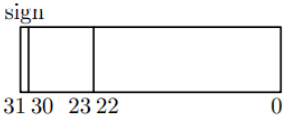


doubleword signed integer

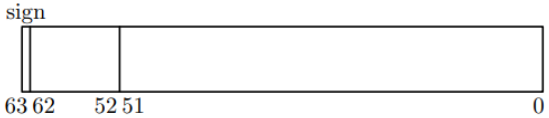


quadword signed integer

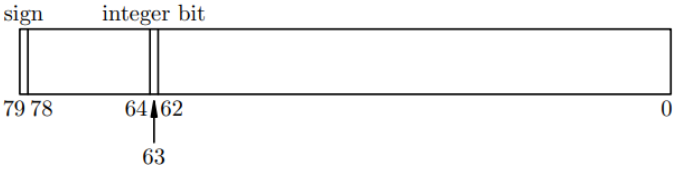
Realni brojevi u floating-point prikazu



single precision floating point



double precision floating point



double extended precision floating point

Prikaz cijelih brojeva, cijeli brojevi bez predznaka

U n bitova možemo spremiti **maksimalno** 2^n različitih podataka (svaki bit može imati vrijednost 0 ili 1). U praksi se iskoriste sve mogućnosti za prikaz stoga skupovi imaju **točno** 2^n elemenata. n je uglavnom potencija broja dva: 8, 16, 32, 64, 128. Najčeće korištene opcije $n = 32$ i $n = 64$.

Cijeli brojevi bez predznaka modeliraju skup $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$. Prikazuje se **najveći mogući** podskup od $\mathbb{N}_0$ koristeći n bitova.

$$Z_{2^n} = \{0, 1, \ldots, 2^{n-2}, 2^{n-1}\}$$

Prikaz proizvoljnog prikazivog broja je **identičan prikazu tog broja u pozicionom zapisu u bazi** 2 **uz dopunu nulama sprijeda** do n bitova.

Cijeli brojevi bez predznaka

Najveći prikazivi cijeli broj za $n \in \{8, 16, 32, 64\}$.

n	$2^n - 1$
8	255
16	65 535
32	4 294 967 295
64	18 446 744 073 709 551 615

Neka je $B \in \{0, 1, \ldots, 2^{n-1}\}$ neki prikazivi broj i neka je

$$B = (b_k b_{k-1} \ldots b_1 b_0)_2 = b_k \cdot 2^k + \cdots + b_1 \cdot 2 + b_0$$

zapis broja B u bazi 2 (**mora biti** $k \leq n - 1$).

Bitovi prikaza broja B kao cijelog broja bez predznaka su:

$$bit_i = \begin{cases} b_i & \text{za } i = 0, \ldots, k \\ 0 & \text{za } i = k+1, \ldots, n-1 \end{cases}$$

Cijeli brojevi bez predznaka - primjer

Neka je $n = 8$, promotrimo zapis broja 120. $120 < 255 = 2^8 - 1$ pa je **broj prikaziv** koristeći 8 bitova memorije.

$120 = 64 + 32 + 16 + 8 =$

$0 \cdot 2^7 + 1 \cdot 2^6 + 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0$

Binarne znamenke su

$b_7 = 0,\ b_6 = 1,\ b_5 = 1,\ b_4 = 1,\ b_3 = 1,\ b_2 = 0,\ b_1 = 0,\ b_0 = 0.$

7	6	5	4	3	2	1	0
0	1	1	1	1	0	0	0

Koristit ćemo i skraćeni zapis: $[bit_{n-1} bit_{n-2} \ldots bit_1 bit_0]$

$120 \leftrightarrow [01111000]$

Aritmetika cijelih brojeva bez predznaka

Aritmetika cijelih brojeva bez predznaka s n bitova za prikaz brojeva je **modularna aritmetika** (**aritmetika ostataka modulo** 2^n).

Aritmetičke operacije $+$, $-$, $\cdot$ na skupu $\mathbb{Z}_{2^n}$ daju rezultat koji je jednak **ostatku** rezultata pripadne **cjelobrojne** operacije (u skupu $\mathbb{Z}$) pri **dijeljenju** s 2^n.

Za prikazive operande A i B vrijedi: `rezultat` $(A\ op\ B) := (A\ op\ B)\ mod\ 2^n$. $op \in \{\oplus_{2^n}, \ominus_{2^n}, \odot_{2^n}\}$.

Računalo **ne javlja grešku** ukoliko je `rezultat`$(A\ op\ B)$ prevelik (neprikaziv) već postavlja bit prijenosa u kontrolnom registru. Ponašanje programa u tom slučaju ovisi o **programu** i **vrsti podataka** koji se prikazuju kao cijeli brojevi bez predznaka:

- Kod korisničkih podataka (brojeva) **prijenos se ignorira** i program radi s rezultatom dobivenim **pravilima modularne aritmetike**.

Aritmetika adresa, modularna aritmetika

- Kod prikazivanja i aritmetike adresa (postoji aritmetika pokazivača) nema ignoriranja registra prijenosa. Javlja se greška, npr. `memory protect violation`.

Brojevi i adrese pripadaju **različitim** tipovima podataka.

Dva bitna razloga realizacije aritmetike cijelih brojeva kao modularna aritmetika:

- **Tehnički:** jednostavna realizacija, brzo izvođenje. Računanje ostatka modulo 2^n se ostvaruje uzimanjem najnižih n bitova rezultata.
- **Matematički:** dobiveni prostor je algebarska struktura prstena ostataka modulo 2^n. **Vrijede poželjna matematička svojstva**: asocijativnost zbrajanja, postojanje neutralnog elementa i suprotnog elementa, komutativnost zbrajanja, asocijativnost množenja, distributivnost množenja u odnosu na zbrajanje.

Dijeljenje cijelih brojeva

Dijeljenje ima smisla na strukturi **polja** (npr. $\mathbb{Q}, \mathbb{R}, \mathbb{C}$). Skupovi $\mathbb{N}_0$, $\mathbb{Z}$, $\mathbb{Z}_{2^n} (n > 1)$ nisu polja (nema inverza za množenje, $\mathbb{N}_0$ niti za zbrajanje).

Rješenje: koristimo dijeljenje s ostatkom (podloga je Euklidov teorem o dijeljenju s ostatkom u skupu $\mathbb{Z}$).

Euklidov teorem: Za svaki cijeli broj $a \in \mathbb{Z}$ i svaki prirodni broj $b \in \mathbb{N}$, postoje jedinstveni brojevi $q, r \in \mathbb{Z}$ takvi da je:
$a = q \cdot b + r,\ 0 \leq r < b$. q je cjelobrojni kvocijent, r je ostatak pri dijeljenju a s b.

Vrijedi $r \in \mathbb{Z}_b := \{0, 1, \ldots, b-1\}$. Uvrštavanjem $b = 2^n$ dobivamo traženi skup $\mathbb{Z}_{2^n}$.

Dijeljenje cijelih brojeva

Rezultati cijelobrojnog dijeljenja:

- $a \ div \ b := q \in \mathbb{Z}$ (operator / u C-u).
- $a \ mod \ b := r \in \mathbb{Z}_b$ (operator % u C-u).

Uz uvjet $r \in \mathbb{Z}_b$ vrijedi $a \ div \ b = q = \left\lfloor \frac{a}{b} \right\rfloor$.

Cjelobrojno dijeljenje s ostatkom vrijedi na domeni $\mathbb{Z} \times \mathbb{N}$.
Cjelobrojno dijeljenje s ostatkom na računalu je **restrikcija** operacija `div` i `mod` ($\mathbb{N}_0 \times \mathbb{N} \subset \mathbb{Z} \times \mathbb{N}$).

Cijeli brojevi bez predznaka u *C*-u

- Cijelim brojevima bez predznaka odgovara tip `unsigned int`, skraćeno `unsigned`.
- Cijelim brojevima s predznakom odgovara tip koji se zove `int`.
- Ovisno o broju bitova korištenim za prikaz, postoji nekoliko veličina: **standardna**, **short**, **long**, ponekad i **long long**.
- Konstante zapisujemo navođenjem vrijednosti i tipa.
- Operacije $+, -, \cdot$ znakovima $+, -, *$.
- Operacije `div` i `mod` znakovima $/, \%$.

Cijeli brojevi bez predznaka u *C*-u, primjer

Primjer cjelobrojnih vrijednosti bez predznaka u C-u

```
1  #include<stdio.h>
2
3  int main(void){
4  unsigned short i = 65535;
5
6  i = i+3;
7  printf("%d\n",i); /*2*/
8  i = i-4;
9  printf("%d\n",i); /*65534*/
10 printf("%d\n",i/10); /*6553*/
11
12 return 0;
13 }
```

$\texttt{USHRT\_MAX} = 65535$ $(n = 16)$.

Cijeli brojevi bez predznaka u *C*-u, primjer

Oprez: %d piše vrijednost uz pretvorbu u tip `int`.
%hu piše vrijednost kao `unsigned short`.

Primjer cjelobrojnih vrijednosti bez predznaka u C-u

```
1 #include<stdio.h>
2
3 int main(void){
4 unsigned short i = 65535;
5 printf("%d\n",(i+3)); /*65538*/
6 printf("%hu\n",(i+3)); /*2*/
7
8 return 0;
9 }
```

Cijeli brojevi s predznakom

- Modeliraju skup cijelih brojeva $\mathbb{Z}$.
- Želimo prikazati **podjednaki dio** negativnih i ne-negativnih brojeva koristeći n bitova.
- Prikazujemo najveći mogući podskup uzastopnih brojeva iz $\mathbb{Z}$ koji je **skoro simetričan** oko 0 (prikazujemo 1 negativni broj više nego pozitivni, puna simetrija bi dala ukupno neparni broj prikazivih elemenata što nije poželjno).
- Od 2^n prikazivih brojeva $2^n/2 = 2^{n-1}$ je **negativno**, isti broj je **ne-negativan** (0 i $2^{n-1} - 1$ pozitivan broj).
- Skup prikazivih cijelih brojeva s predznakom je:

$$\mathbb{Z}_{2^n}^{-} = \{-2^{n-1}, -2^{n-1}+1, \ldots, -2, -1, 0, 1, 2, \ldots, 2^{n-1}-2, 2^{n-1}-1\}$$

Cijeli brojevi s predznakom

Vrijednosti granica za različiti n:

n	-2^{n-1}	$2^{n-1}-1$
8	-128	127
16	$-32\ 768$	32 767
32	$-2\ 147\ 483\ 648$	2 147 483 647

S obzirom na mali raspon prikazivih brojeva sve više se koristi $n = 64$. $2^{63} - 1 = 9\ 223\ 372\ 036\ 854\ 775\ 807$.

Prikaz pojedinog (prikazivog) broja s predznakom je određen s dva uvjeta:

- Nenegativni brojevi moraju imati isti prikaz kao brojevi bez predznaka.
- Aritmetika za te prikaze mora dati dobru algebarsku strukturu na $\mathbb{Z}_{2^n}^{-}$ (zatvorenost zbrajanja i množenja, dobra svojstva tih operacija).

Prikaz cijelih brojeva s predznakom

$\mathbb{Z}_{2^n}^-$ je potpuni sustav ostataka modulo 2^n. $\mathbb{Z}_{2^n}^-$ uz operacije $+$ i $\cdot$ tvori prsten s jedinicom.

Da bi dobili prikaz svih nenegativnih cijelih brojeva definiramo:

- iste prikaze imaju oni brojevi koji imaju jednaki pravi ostatak modulo 2^n u skupu $\mathbb{Z}_{2^n}$ i u skupu $\mathbb{Z}_{2^n}^-$.
- Za $B = 1, \ldots, 2^{n-1}$, prikaz negativnog broja $-B$ jednak je prikazu broja $2^n - B$ (bez predznaka).

Prikaz cijelih brojeva s predznakom - primjer

Usporedba prikaza brojeva u $\mathbb{Z}_8$ i $\mathbb{Z}_8^-$ ($n = 3$).

$B \in \mathbb{Z}_8$	$B \in \mathbb{Z}_8^-$	prikaz
0	0	000
1	1	001
2	2	010
3	3	011
4	-4	100
5	-3	101
6	-2	110
7	-1	111

Primjetimo: $0 - 1 = [000] - [001] = [111] = -1$,
$-1 + 1 = [111] + [001] = [000]$ (**zbog modularne aritmetike**).
$-4 + 1 = [100] + [001] = [101] = -3$,
$-4 - 1 = [100] - [001] = [011] = 3$,
$3 + 1 = [011] + [001] = [100] = -4$.

Prikaz cijelih brojeva s predznakom - primjer

Prethodno uočena svojstva vrijede generalno:
$-2^{n-1} \leftrightarrow 2^n - 2^{n-1} = 2^{n-1} = [100\ldots00]$.
$-1 \leftrightarrow 2^n - 1 = [111\ldots111]$.
$((2^{n-1} - 1) + 1) = ((2^{n-1} - 1) + 1)\ mod\ 2^n = 2^{n-1}\ mod\ 2^n = -2^{n-1} \in \mathbb{Z}_{2^n}^{-}$ $(2^{n-1} \notin \mathbb{Z}_{2^n}^{-})$. Isto dobijemo i binarnim zbrajanjem:
$[011\ldots111] + [000\ldots001] = [100\ldots000]$.
$-1 + 1 = [111\ldots111] + [000\ldots001] = (1)[000\ldots000] = 0$.
Prijenos se modularno ignorira.

Primjetimo: svi negativni brojevi imaju početni bit vrijednosti 1 dok svi ne-negativni brojevi imaju početni bit vrijednosti 0.

Prikaz suprotnog broja preko komplementa

Za veći n nije jednostavno računati $2^n - B$ da bi pronašli prikaz nekog broja $-B$, gdje $B > 0$.

Možemo koristiti jednostavniji postupak, **tehniku dvojnog komplementa**.

- Prikaz broja B treba **komplementirati**, pretvoriti bitove vrijednosti 1 u 0 i bitove vrijednosti 0 u 1 ($0 \leftarrow 1$).
- Komplementiranom prikazu treba dodati 1 modulo 2^n.

Zašto to radi?

Za broj B i komplementirani prikaz $\bar{B}$ vrijedi:

$B + \bar{B} = [111\ldots111] = 2^n - 1$

$\bar{B} + 1 = 2^n - B$ što je prikaz broja $-B$.

To možemo pokazati i u modularnoj aritmetici:

$B + \bar{B} + 1 = 2^n$

$B \oplus_{2^n} (\bar{B} \oplus_{2^n} 1) = 0$ (jedinstveni inverz broja B s obzirom na operaciju zbrajanja $\oplus_{2^n}$ jednak je $\bar{B} \oplus_{2^n} 1$).

Prikaz cijelih brojeva s predznakom - primjer

Pronađimo zapis broja -110 u $n = 8$ bitova. Vrijedi:
$-2^7 = -128 \leq -110 \leq 127 = 2^7 - 1$, stoga je -110 prikaziv.
Binarni prikaz broja $B = 110$ je:
$110 = 64 + 32 + 8 + 4 + 2 =$
$0 \cdot 2^7 + 1 \cdot 2^6 + 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0$.
110 kao cijeli broj bez predznaka ima prikaz: $110 \leftrightarrow [01101110]$.

7	6	5	4	3	2	1	0
0	1	1	0	1	1	1	0

Komplementiranjem dobivamo:

7	6	5	4	3	2	1	0
1	0	0	1	0	0	0	1

Prikaz cijelih brojeva s predznakom - primjer

Dodamo 1 modulo 2^8 i dobijemo prikaz broja -110.

7	6	5	4	3	2	1	0
1	0	0	1	0	0	1	0

Identičan rezultat možemo dobiti računanjem
$2^8 - 110 = 256 - 110 = 146$.
$146 = 128+16+2 = 1\cdot 2^7+0\cdot 2^6+0\cdot 2^5+1\cdot 2^4+0\cdot 2^3+0\cdot 2^2+1\cdot 2^1+0\cdot 2^0$.
Koristeći 8 bitova, $-110 \leftrightarrow 146 \leftrightarrow [10010010]$.

Dijeljenje cijelih brojeva - izbor ostatka

Ograničenje na ostatak $0 \leq r < b$, tj. $r \in \mathbb{Z}_b$ prirodno odgovara cijelim brojevima bez predznaka. Zbog toga računanje funkcionira prema standardnom Euklidovom teoremu samo na $\mathbb{N}_0 \cup \mathbb{N}$.

Kod **negativnih** brojeva možda ima smisla dozvoliti i da ostatak bude **negativan**. Izbor ovisi o primjeni i željenim svojstvima rezultata.

Standard $C90$ ne propisuje način računanja ostatka (ovisi o implementaciji).
Način računanja je precizno definiran u verziji standarda $C99$.

Dijeljenje cijelih brojeva s predznakom prema $C99$

- kvocijent se uvijek zaokružuje prema nuli:

$$q = sign(\frac{a}{b}) \cdot \lfloor |\frac{a}{b}| \rfloor$$

- ostatak ima isti predznak kao a

$$r = sign(a) \cdot (|a| mod |b|)$$

Za ostatak r vrijedi:

- ako je $a \geq 0$ onda $r \in \mathbb{Z}_b,\ 0 \leq r < |b|$.
- ako je $a < 0$, onda $r \in -\mathbb{Z}_b,\ -|b| < r \leq 0$.

Prednosti ovakve definicije:

- Dobivamo iste apsolutne vrijednosti kvocijenta q i ostatka r bez obzira na predznake od a i b.
- Samo predznaci od q i r ovise o predznacima od a i b.

Cijeli brojevi s predznakom u *C*-u

Cijelim brojevima s predznakom odgovara tip `int`.
Prema broju bitova koji se koriste za prikaz postoje **standardni** `int` ($n = 32$), `short` ($n = 16$), long ($n = 32$), ponekad i long long ($n = 64$).

Primjer cjelobrojnih brojeva s predznakom u C-u

```
1  #include<stdio.h>
2  /*SHRT_MAX = 32767, n=16, (limits.h)*/
3  int main(void){
4  short int i = 32766;
5  i+=1;
6  printf("%d\n",i);/*32767*/
7  i+=1;
8  printf("%d\n",i);/*-32768*/
9  return 0;
10 }
```

Modularna aritmetika kod dodijeljivanja u C-u

```
1  #include<stdio.h>
2
3  int main(void){
4  int broj;
5  broj = 10; printf("␣broj␣=␣%d\n",broj);
6  broj = 2100000000; printf("␣broj␣=␣%d\n",broj);
7  broj = 7000000000; printf("␣broj␣=␣%d\n",broj);
8  broj = 9000000000; printf("␣broj␣=␣%d\n",broj);
9  return 0;
10 }
```

Izlaz programa:

broj $= 10$

broj $= 2100000000$

broj $= -1589934592$

broj $= 410065408$

Modularna aritmetika kod učitavanja u C-u

```
1 #include<stdio.h>
2
3 int main(void){
4 int broj;
5
6 scanf("%d",&broj);
7 printf("Ucitani broj = %d\n",broj);
8 return 0;
9 }
```

Za ulaz 9000000000, program ispiše vrijednost 410065408.

Cijeli brojevi - klasične pogreške

Računanje $n!$ u cjelobrojnoj aritmetici.

Vrijedi:

$1! = 1$

$n! = n \cdot (n-1)!$

$n! = 1 \cdot 2 \cdot \dots \cdot (n-1) \cdot n$

Računanje 50! u C-u

```
1 #include <stdio.h>
2
3 int main(void){
4 int i,f50 = 1; /*n = 32 za int*/
5 //f50 - inicijalizacija na neutral za mnozenje
6 for(i=2;i<=50;i++)  f50*=i;
7 printf("f50 = %d\n",f50); /*f50=0*/
8 return 0;
9 }
```

Cijeli brojevi - klasične pogreške

Zašto je izlaz programa $f_{50} = 0$?

$50! = 30414\ 09320\ 17133\ 78043\ 61260\ 81660$
$64768\ 84437\ 76415\ 68960\ 51200\ 00000\ 00000$ $n!$ ima 65 znamenki i **nije prikaziv** u cjelobrojnoj aritmetici. Pošto je cjelobrojna aritmetika modularna, rezultat sugerira da je $50! = 0 \ mod\ 2^{32}$. Odnosno 2^{32} dijeli $50!$.

Zadatak: Nađite najveću potenciju broja 2 koja dijeli $n!$.

Rješenje: Očito nije dobro rješenje računati $n!$ te zatim dijeljenjem sa 2 tražiti najveću potenciju. Razlog - modularna aritmetika, nemogućnost prikaza $n!$ već i za relativno mali n.

Umjesto toga potenciju možemo izračunati analitički:

$$m = \lfloor \frac{n}{2} \rfloor + \lfloor \frac{n}{2^2} \rfloor + \lfloor \frac{n}{2^3} \rfloor + \lfloor \frac{n}{2^4} \rfloor + \lfloor \frac{n}{2^5} \rfloor \dots$$

Za $n = 50$, $m = 25 + 12 + 6 + 3 + 1 = 47$.

Cijeli brojevi - primjer

Zadatak: Pronađite najmanji prirodni broj n za koji je $n! = 0$ u cijelobrojnoj aritmetici s l bitova za prikaz brojeva (mod 2^l).

Rješenje zadatka

```
1  #include<stdio.h>
2
3  int main(void){
4  short broj1 = 1; /*(l=16)*/
5  int broj2 = 1,i, minb1 = 0, minb2 = 0; /*l=32*/
6  for(i=2;;i++){
7    if(!minb1) broj1*=i; if(!minb2) broj2*=i;
8      if(broj1 == 0 && minb1 == 0) minb1 = i;
9      if(broj2 == 0 && minb2 == 0) minb2 = i;
10     if(minb1!=0 && minb2!=0) break; }
11 printf("%d, %d\n",minb1, minb2);/*18, 34*/
12 return 0; }
```