Programiranje 1
3. predavanje

Saša Singer

singer@math.hr
web.math.pmf.unizg.hr/~singer

PMF – Matematički odsjek, Zagreb

Sadržaj predavanja

- Osnovni tipovi podataka u računalu (pregled):
  - Višekratnici byte-a, odn. riječi (bez strukture).
  - Znakovi, logičke vrijednosti.
  - Cijeli brojevi bez predznaka i s predznakom.
  - "Realni" (floating–point) brojevi.
- Cijeli brojevi — prikaz i aritmetika:
  - Prikaz brojeva bez predznaka — sustav ostataka.
  - Modularna aritmetika cijelih brojeva.
  - Dijeljenje cijelih brojeva — kvocijent i ostatak.
  - Prikaz brojeva s predznakom — sustav ostataka, dijeljenje.
  - Tipične pogreške u korištenju cijelih brojeva.

Informacije — odrada

Termin odrade predavanja 25. 1. 2019. je:

- subota, 20. 10., od 10–12 u (003).

Prikaz podataka u računalu

Sadržaj

- Osnovni tipovi podataka u računalu (pregled):
  - Višekratnici byte-a, odn. riječi (bez strukture).
  - Znakovi, logičke vrijednosti.
  - Cijeli brojevi bez predznaka.
  - Cijeli brojevi s predznakom.
  - "Realni" (floating–point) brojevi.

Osnovni tipovi podataka u računalu

Jednostavno rečeno, osnovni ili fundamentalni tipovi podataka u računalu su:

- one cjeline ili blokovi bitova s kojima računalo "zna nešto raditi", i to neovisno o njihovom sadržaju.

To znači da postoje instrukcije koje nešto rade s tim cjelinama kao operandima, bez obzira na eventualni dodatni tip operanda. U ovom kontekstu je

- tip = interpretacija (odnosno, značenje) sadržaja.

Tipični primjer takvih instrukcija su

- instrukcije za transfer tih cjelina između memorije i registara procesora.

Osnovni tipovi podataka (nastavak)

Ako se sjetimo "pravokutnog" izgleda memorije, onda u te tipove sigurno ulaze

- osnovne cjeline koje možemo adresirati,

dakle, ono što smo ranije (u skici memorije) nazvali riječ/byte.

Napomena: kasnije smo se dogovorili da pojam "riječ" znači nešto drugo — prostor za spremanje cijelih brojeva.

Osim toga, ovisno o veličini "osnovne stranice" (jedne adrese) memorije, računalo može "znati" raditi i s

- manjim cjelinama — dijelovima osnovne cjeline, ako je ona dovoljno velika (jedan byte se ne dijeli),
- većim cjelinama — blokovima osnovnih cjelina.

Osnovni tipovi podataka (nastavak)

Vidimo da ti osnovni tipovi vrlo ovise o arhitekturi računala.

Obzirom na to da sadržaj nije bitan, zapravo, jedino što se može reći o tim cjelinama je

- njihova duljina — u bitovima.

Naravno, imena ili nazivi za cjeline istih duljina variraju na raznim arhitekturama.

Za nas kao korisnike, ovi tipovi nisu naročito važni, ali

- zgodno ih je upoznati (navesti njihove duljine i nazive), barem na jednom primjeru.

Primjer. Intelova 32-bitna arhitektura procesora (IA–32).

Osnovni tipovi podataka na IA–32

Osnovna cjelina koju možemo adresirati na IA–32 je

1 byte = 8 bitova.

To je duljina "osnovne stranice" (jedne adrese) memorije.

Ostali osnovni tipovi podataka na IA–32 su većih duljina:

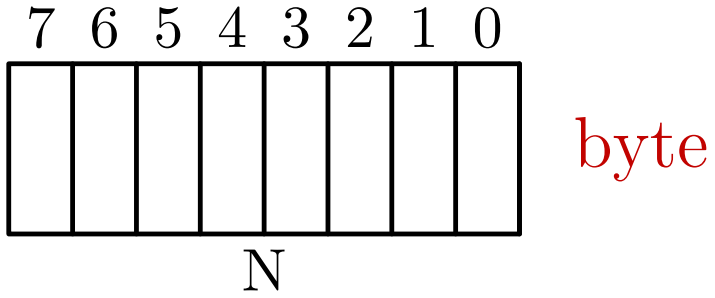
Naziv	Duljina (bitova)	Broj byteova
word	16	2
doubleword	32	4
quadword	64	8
double quadword	128	16

Označavanje bitova i adresa

Na primjeru osnovnih tipova podataka zgodno je odmah uvesti još dvije stvari:

- standardne oznake za bitove unutar pojedine cjeline (koriste se općenito, a ne samo na IA–32),
- način adresiranja pojedinih dijelova cjeline, obzirom na raspored bitova (specifično za pojedinu arhitekturu).

Jedan byte, duljine $n = 8$ bitova, na adresi N, označavamo ovako (svaka "kućica" je 1 bit):



Označavanje bitova i adresa (nastavak)

Objašnjenje oznaka: Uzmimo da neka cjelina (u ovom slučaju, osnovni tip podataka) ima duljinu od n bitova.

- Tradicionalno se pojedini bitovi u cjelini indeksiraju slično kao i riječi u memoriji, dakle, od 0 do $n-1$.

Međutim, ovdje poredak ide "naopako", tako da

- "najdesniji" bit ima indeks 0 — najniži ili zadnji bit,
- "najljeviji" bit ima indeks $n-1$ — najviši ili vodeći bit.

Razlog: pozicioni prikaz cijelih brojeva (u bazi 2), u kojem vodeću znamenku (bit) pišemo kao prvu (lijevu), a najnižu znamenku (bit) pišemo kao zadnju (desnu). Osim toga, indeksi bitova odgovaraju pripadnim potencijama baze 2.

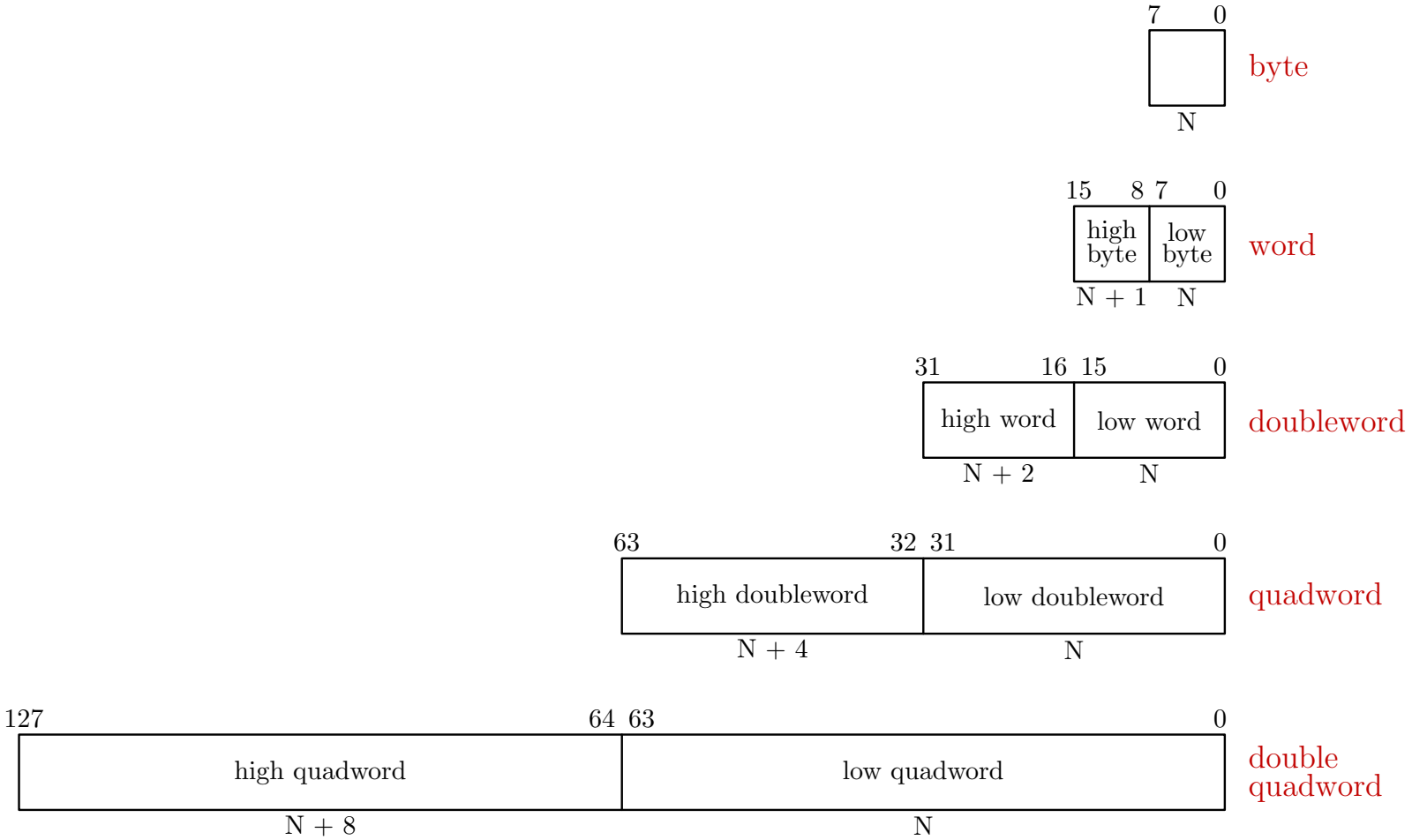
Detalji malo kasnije!

Označavanje bitova i adresa (nastavak)

- Ove oznake za bitove katkad pišemo iznad, a katkad ispod skice cjeline (ali značenje je uvijek očito iz slike).
- Na početku ih pišemo iznad, jer ispod skice pišemo adrese na kojima se nalaze pojedini adresabilni dijelovi cjeline — u ovom slučaju, to su byteovi (adresabilne cjeline na IA–32 su byteovi).
- Na IA–32, najniži byteovi su i na najnižim adresama (tzv. "little endian" encoding),
  ali postoje arhitekture računala na kojima je obratno, tj. vodeći dio je na najnižoj adresi (tzv. "big endian").
- Dogovorno, najniža adresa — na kojoj "počinje" cjelina, je, ujedno, i adresa čitave cjeline (bez obzira na poredak dijelova). Ovo se standardno koristi u C-u.

Osnovni tipovi podataka na IA–32 (nastavak)

Osnovni tipovi podataka na IA–32 onda izgledaju ovako:



Jednostavni tipovi podataka

Ponovimo, ovi osnovni tipovi podataka nisu jako korisni, jer s njima ne možemo ništa "pametnije" raditi, osim transfera.

Za stvarno "računanje", trebamo

- dodatnu interpretaciju sadržaja (bitova) cjeline i
- operacije s takvom vrstom podataka (cjelinom bitova).

Skup podataka i operacije na njima čine neku algebarsku strukturu, koju zajedničkim imenom zovemo tip podataka.

Oni tipovi podataka za koje računalo "zna" ili može:

- prikazati pripadni skup podataka i
- izvesti pripadne operacije na njima,

zovu se jednostavni tipovi podataka.

Jednostavni tipovi podataka (nastavak)

Pojam "jednostavni" znači da su operacije na toj vrsti podataka izravno podržane arhitekturom računala, tj.

- postoje instrukcije za njih.

Dakle, te operacije su elementarne operacije (za računalo kao izvršitelja) i, u principu, su brze.

Standardne jednostavne tipove podataka možemo grubo podijeliti u dvije grupe:

- nenumerički tipovi — znakovi, logička algebra,
- numerički tipovi — "cijeli" i "realni" brojevi (nekoliko raznih tipova za obje vrste brojeva).

Vidjet ćemo da se nenumerički tipovi, zapravo, svode na numeričke.

Nenumerički tipovi podataka (pregled)

Ukratko o nenumeričkim tipovima podataka.

Znakovi:

- prikaz je u nekom kôdu (obično, nadskup ASCII kôda), tj. cijelim brojevima,
- posebnih operacija sa znakovima (osim transfera) nema. Sve funkcije na znakovima svode se na elementarne operacije na cijelim brojevima (vidi C).

Dakle, znakovi nisu izravno vezani za arhitekturu računala, ali su jednostavni tip u operacijskim sustavima i programskim alatima.

Uglavnom, služe za "humani" ulaz i izlaz, te kao dijelovi složenijih struktura podataka.

Znakovi (primjer)

Znakovni tip u programskom jeziku C zove se `char`.

```
#include <stdio.h>

int main(void) {
    char c = '1';

    printf("%c\n", c);  /* 1 */
    printf("%d\n", c);  /* 49 */

    return 0;
}
```

`%c` — piše vrijednost kao znak (`char`),
`%d` — piše vrijednost kao cijeli broj (`int`), i to decimalno.

Nenumerički tipovi podataka (pregled)

Logička ili Booleova algebra:

- logičke vrijednosti prikazuju se bitovima,

$$\text{laž} = 0, \quad \text{istina} = 1,$$

  koje, opet, možemo uzeti i kao cijele brojeve,
- osnovne operacije ne, i, ili (engl. not, and, or), svode se na aritmetičke operacije u bazi 2.

Logičke operacije mogu se izvesti i bit–po–bit na čitavim skupinama bitova u nekoj većoj cjelini (vidi C).

Za nas kao korisnike, logička algebra služi za formulaciju i kombiniranje uvjeta u uvjetnim naredbama i petljama.

Logičke vrijednosti (primjer)

C nema posebni tip za logičke vrijednosti, već izravno koristi cijele brojeve: laž = 0, istina = 1.

```
#include <stdio.h>

int main(void) {
    int i = 10, j = 20;

    printf("%d\n", i < j);    /* 1 */
    printf("%d\n", i >= j);   /* 0 */
    printf("%d\n", i == j);   /* 0 */

    return 0;
}
```

Numerički tipovi podataka (pregled)

Numerički tipovi podataka moraju realizirati

- četiri osnovne aritmetičke operacije na raznim skupovima brojeva.

Osnovni problem: standardni skupovi brojeva u matematici

$$\mathbb{N}, \quad \mathbb{Z}, \quad \mathbb{Q}, \quad \mathbb{R}$$

su beskonačni i ne možemo ih prikazati u računalu.

Umjesto toga, u računalu možemo prikazati samo neke konačne podskupove odgovarajućeg matematičkog skupa. Drugim riječima,

- konačni podskup je "model" beskonačnog skupa.

Numerički tipovi podataka (pregled)

Numeričke tipove možemo podijeliti u tri grupe, prema beskonačnom skupu kojeg "modeliramo":

- "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 $\mathbb{R}$.

Navodnici naglašavaju da su pripadni "prikazivi" skupovi brojeva konačni.

Dodatno, svaka grupa ima nekoliko "podtipova", ovisno o "veličini" pripadnog konačnog skupa prikazivih brojeva, tj.

- ovisno o broju bitova predviđenom za prikaz.

Numerički tipovi podataka (pregled)

Osim toga, prijelaz na konačne skupove bitno mijenja realizaciju aritmetike na odgovarajućem skupu. Aritmetika se ne nasljeđuje projekcijom s originalnog skupa!

Za potpuni opis numeričkih tipova podataka, moramo još opisati:

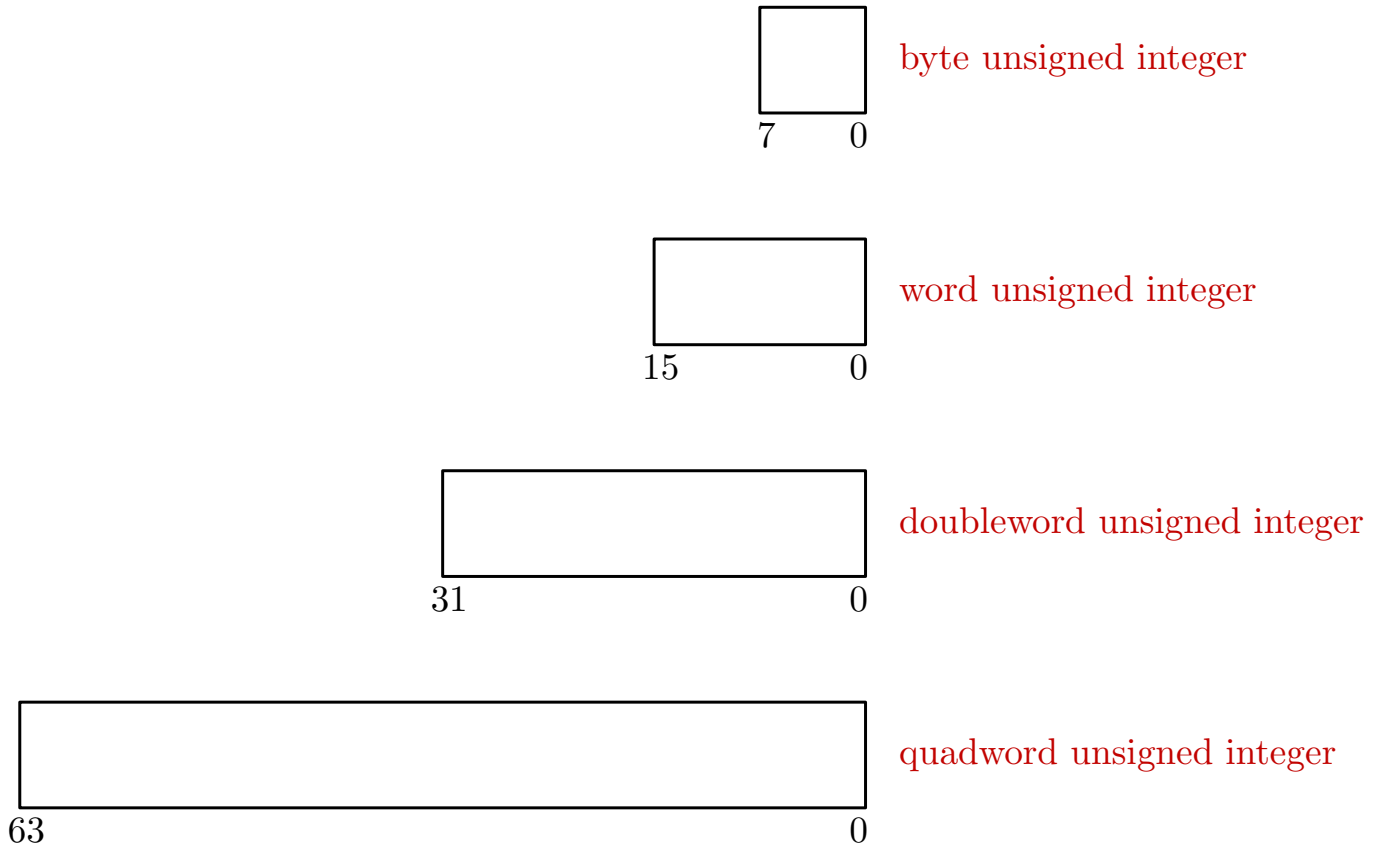
- koji konačni skupovi brojeva modeliraju odgovarajuće matematičke skupove,
- kako se točno prikazuju njihovi elementi u računalu,
- kako se realizira artimetika na tim skupovima.

Detalji u nastavku.

Prije toga, za ilustraciju, pogledajmo kako izgledaju ove tri grupe numeričkih tipova podataka (s podtipovima) na IA–32 arhitekturi.

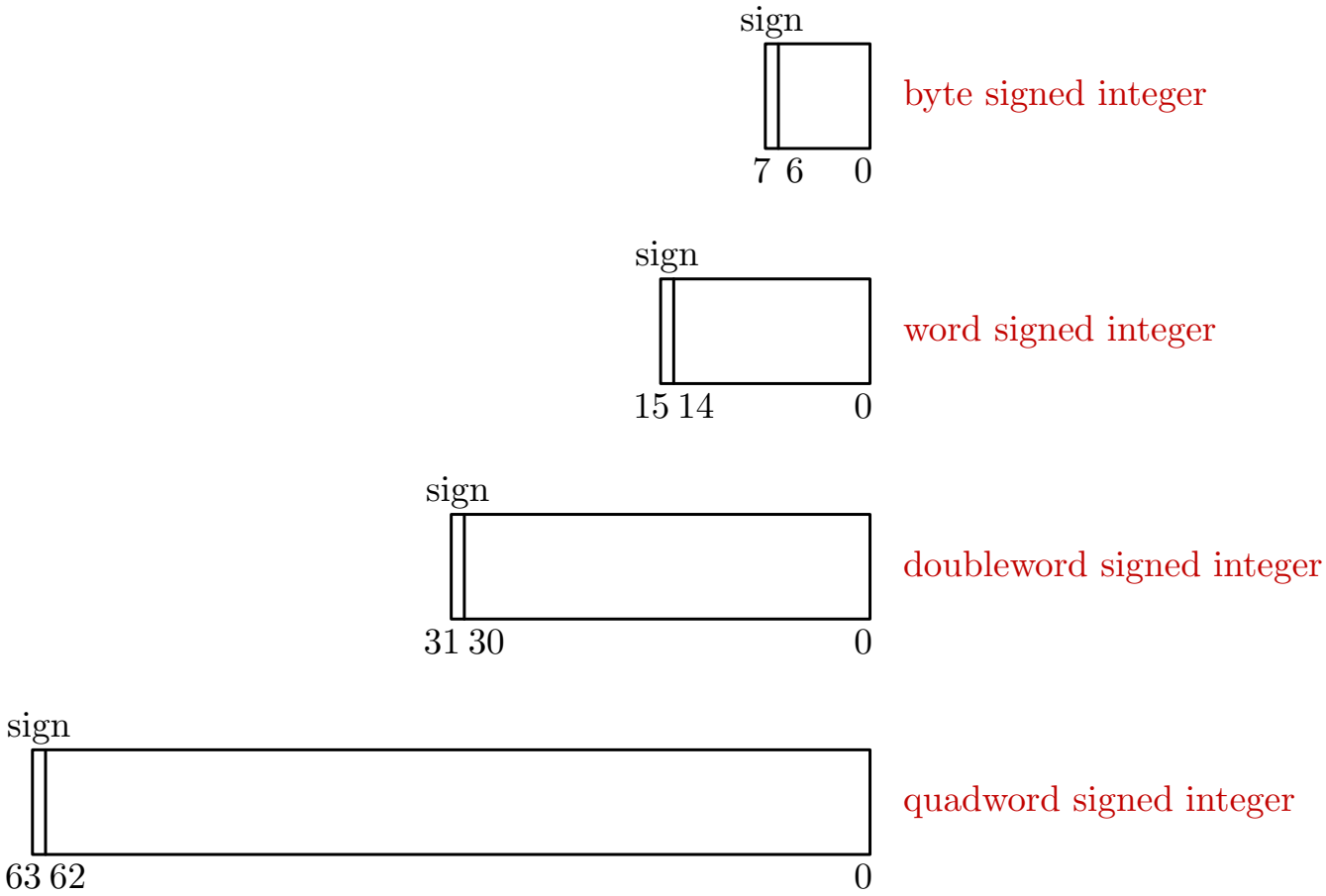
Numerički tipovi podataka na IA–32

Cijeli brojevi bez predznaka (unsigned integer) na IA–32:



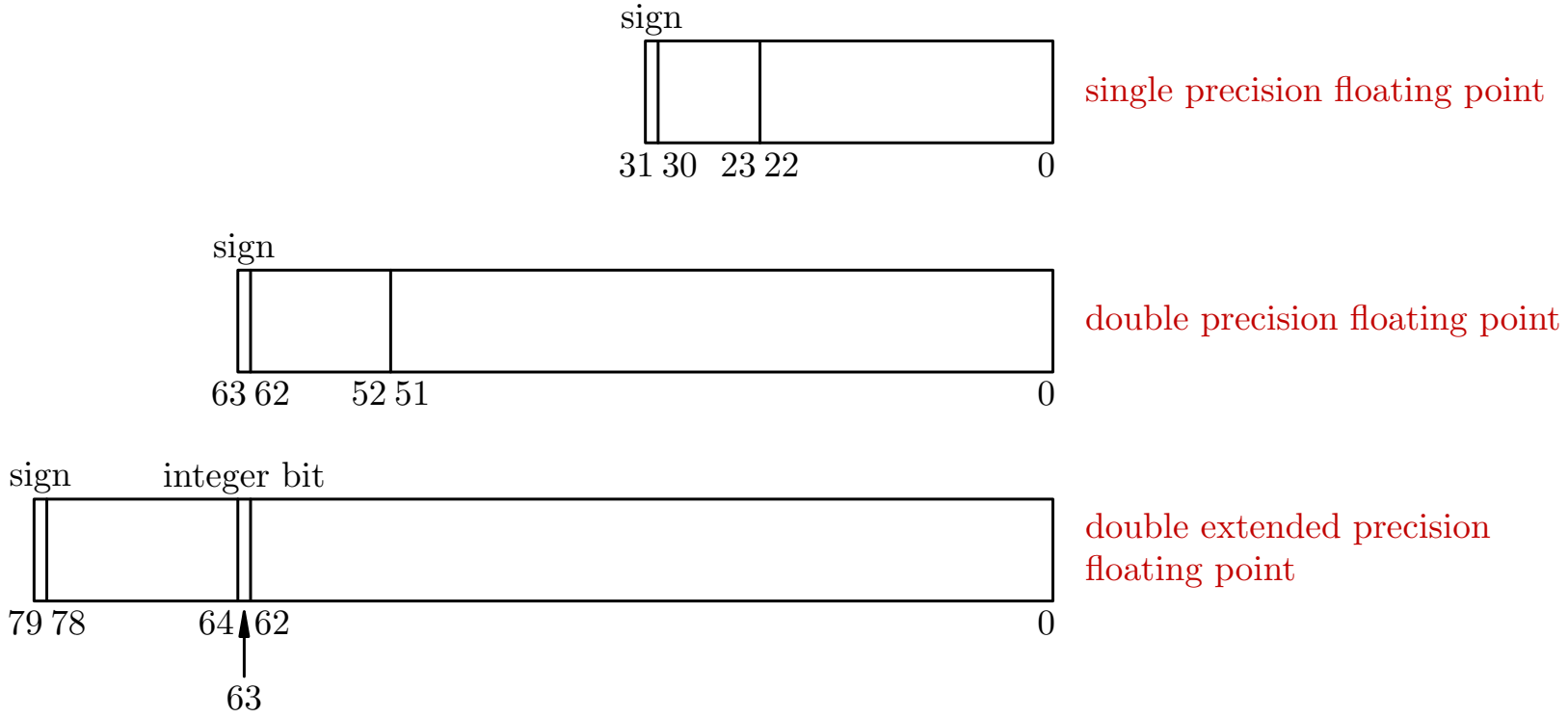
Numerički tipovi podataka na IA–32 (nastavak)

Cijeli brojevi s predznakom (signed integer) na IA–32:



Numerički tipovi podataka na IA–32 (nastavak)

Realni brojevi u tzv. "floating–point" prikazu (v. kasnije), ne samo na IA–32, već općenito, po IEEE standardu:



Skraćeni nazivi za ove tipove su: single, double i extended.

Prikaz cijelih brojeva u računalu

Sadržaj

- Cijeli brojevi — prikaz i aritmetika:
  - Prikaz brojeva bez predznaka.
  - Sustav ostataka, prsten ostataka modulo 2^n.
  - Modularna aritmetika cijelih brojeva.
  - Aritmetika adresa — "obična" aritmetika.
  - Dijeljenje cijelih brojeva — Euklidov teorem, cjelobrojni kvocijent i ostatak.
  - Prikaz brojeva s predznakom — sustav ostataka.
  - Prikaz negativnih brojeva — komplementiraj i dodaj 1.
  - Dijeljenje cijelih brojeva s predznakom.
  - Tipične pogreške u korištenju cijelih brojeva.

Uvod u prikaz cijelih brojeva

Prvo i osnovno što moramo znati je broj bitova predviđenih za prikaz cijelih brojeva (bez predznaka ili s njim).

Pretpostavimo da imamo n bitova na raspolaganju za prikaz. Već smo vidjeli da su tipične vrijednosti za n

$$8, \quad 16, \quad 32, \quad 64, \quad 128.$$

Danas se (još uvijek) najčešće koristi $n = 32$.

U n bitova možemo prikazati točno 2^n različitih podataka, jer svaki bit može biti jednak 0 ili 1, i to nezavisno od ostalih.

Dakle, skup brojeva koje možemo prikazati u tih n bitova ima (najviše) 2^n elemenata. Običaj je da se iskoriste sve mogućnosti za prikaz, pa taj skup ima točno 2^n elemenata.

Cijeli brojevi bez predznaka

Cijeli brojevi bez predznaka modeliraju skup $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$.

Standardni dogovor: u računalu se prikazuje

- najveći mogući početni komad tog skupa $\mathbb{N}_0$.

Ako imamo n bitova na raspolaganju za prikaz, onda skup svih prikazivih brojeva ima 2^n elemenata, pa je on jednak

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

Dakle, najveći prikazivi cijeli broj bez predznaka je

$$2^n - 1.$$

Veće brojeve ne možemo prikazati koristeći samo n bitova.

Cijeli brojevi bez predznaka (nastavak)

Tipične vrijednosti za najveći prikazivi cijeli broj bez predznaka su:

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

Kako stvarno izgleda prikaz brojeva u tih n bitova?

Prikaz pojedinog (prikazivog) broja je

- doslovna "kopija" prikaza tog broja u pozicionom zapisu u bazi 2, uz dopunu nulama "sprijeda" do n bitova.

Što to znači?

Prikaz cijelih brojeva bez predznaka

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

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

pozicioni zapis tog broja B u bazi 2, gdje su $b_i \in \{0, 1\}$ njegove binarne znamenke (bitovi). Uočiti: B prikaziv $\iff k \leq n - 1$.

I točno tako se spremaju bitovi u prikazu, samo treba vodeće bitove dopuniti nulama, od indeksa $k + 1$ do $n - 1$, ako takvih ima, tj. ako je $k < n - 1$.

Dakle, prikaz broja B kao cijelog broja bez predznaka ima oblik

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

Prikaz cijelih brojeva bez predznaka — primjer

Primjer. Uzmimo da je $n = 8$ (budimo skromni) i pogledajmo zapis broja 123. Za početak, vrijedi

$$123 \leq 255 = 2^8 - 1,$$

pa je 123 prikaziv. Nadalje, njegov prošireni binarni prikaz je

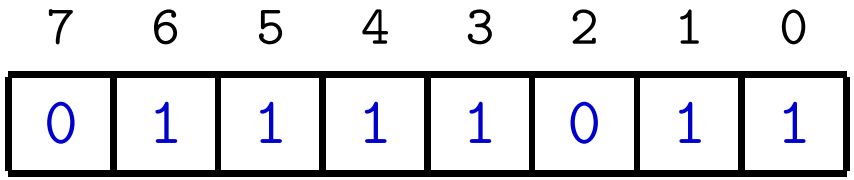
$$\begin{aligned} 123 &= 64 + 32 + 16 + 8 + 2 + 1 \\ &= 0 \cdot 2^7 + 1 \cdot 2^6 + 1 \cdot 2^5 + 1 \cdot 2^4 \\ &\quad + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0. \end{aligned}$$

Binarne znamenke broja 123 su

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

Prikaz cijelih brojeva bez predznaka — primjer

Dakle, broj 123, kao cijeli broj bez predznaka, ima prikaz



Katkad je zgodno imati i skraćenu oznaku

$$[\ \mathrm{bit}_{n-1}\ \mathrm{bit}_{n-2}\ \mathrm{bit}_{n-3}\ \ldots\ \mathrm{bit}_1\ \mathrm{bit}_0\ ]$$

za zapis vrijednosti svih n bitova u prikazu broja.

Broj 123, kao cijeli broj bez predznaka, onda ima prikaz

$$123 \quad \longleftrightarrow \quad [\ 0\ 1\ 1\ 1\ 1\ 0\ 1\ 1\ ]\ .$$

Napomena. Opći algoritam za nalaženje binarnih znamenki broja napravit ćemo kasnije.

Aritmetika cijelih brojeva bez predznaka

Aritmetika cijelih brojeva bez predznaka s n bitova za prikaz brojeva je tzv. modularna aritmetika, ili, preciznije

- aritmetika ostataka modulo 2^n.

To znači da aritmetičke operacije $+$, $-$ i $*$, na skupu $\mathbb{Z}_{2^n}$ cijelih brojeva bez predznaka, daju rezultat koji je

- jednak ostatku rezultata pripadne cjelobrojne operacije (u skupu $\mathbb{Z}$) pri dijeljenju s 2^n.

Drugim riječima, za prikazive operande A i B vrijedi

$$\text{rezultat } (A \text{ op } B) := (A \text{ op } B) \bmod 2^n,$$

gdje je op zbrajanje, oduzimanje ili množenje.

Oznake za pripadne operacije u matematici su $\oplus_{2^n}$, $\ominus_{2^n}$ i $\odot_{2^n}$.

Aritmetika cijelih brojeva bez predznaka

Važna napomena. Ako je "pravi" cjelobrojni rezultat A op B prevelik (neprikaziv), računalo ne javlja nikakvu grešku, već

- postavlja tzv. bit prijenosa (engl. "carry bit") na 1 u kontrolnom registru.

Što će se dalje dogoditi, ovisi o programu koji se izvršava.

Standardno ponašanje programskih alata ovisi o tome koja vrsta podataka se prikazuje cijelim brojevima bez predznaka.

A tu postoje dvije mogućnosti.

- Ako je riječ o korisničkim podacima, tj. o "brojevima", prijenos se ignorira, bez ikakve poruke. Normalno se nastavlja rad, s rezultatom po opisanom pravilu.

Zato oprez (v. malo kasnije)!

Aritmetika adresa

S druge strane, memorijske adrese se, također, prikazuju kao cijeli brojevi bez predznaka — određene veličine.

Na primjer, obični adresni prostor na IA–32 je 2^{32} byte-a, pa se adrese prikazuju kao 32–bitni cijeli brojevi bez predznaka.

- S adresama se isto tako rade aritmetičke operacije,
  - tzv. aritmetika pokazivača ili pointera u C-u, posebno kod obrade nizova (stiže u Prog2).
- Dosta je očito da ova aritmetika nije modularna!
- Tu nema šale, prijenos se ne smije ignorirati i računalo mora javiti grešku ("memory protect violation" ili nešto slično).

Zato brojevi i adrese pripadaju različitim tipovima podataka!

Modularna aritmetika — opravdanje

Zašto se aritmetika cijelih brojeva realizira na ovaj način,

- kao modularna aritmetika?

Postoje dva vrlo dobra razloga.

- Čisto tehnički, ova realizacija je najlakša i brza.

Ostaci modulo 2^n (uz ignoriranje prijenosa) znače da stalno uzimamo samo najnižih n bitova rezultata.

Drugi razlog je matematičke prirode.

- U pozadini ove realizacije je klasična algebarska struktura prstena ostataka modulo 2^n (v. Dodatak).

Tu strukturu je korisno detaljnije upoznati, jer bitno olakšava razumijevanje cjelobrojne aritmetike (i one s predznakom).

Dijeljenje cijelih brojeva

A što je s dijeljenjem? To dosad nismo ni spomenuli!

S razlogom! "Obično" dijeljenje ima smisla tek u strukturi polja, poput racionalnih brojeva $\mathbb{Q}$, ili realnih brojeva $\mathbb{R}$.

Znamo da $\mathbb{N}_0$ i $\mathbb{Z}$ nisu polja. Isto vrijedi i za $\mathbb{Z}_{2^n}$, čim je $n > 1$, a slučaj $n = 1$ (tj. polje $\mathbb{Z}_2$) je potpuno neinteresantan za praksu, barem što se tiče aritmetike cijelih brojeva (iako je vrlo bitan za logičku algebru).

Što sad?

Zamjena za "obično" dijeljenje u cijelim brojevima je tzv. dijeljenje s ostatkom. Podloga za to je poznati Euklidov teorem o dijeljenju s ostatkom u skupu $\mathbb{Z}$.

Dijeljenje cijelih brojeva — Euklidov teorem

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

$$a = q \cdot b + r \quad \text{i} \quad 0 \leq r < b.$$

Broj q je cjelobrojni kvocijent, a r ostatak pri dijeljenju a s b.

Ograničenje $0 \leq r < b$ znači da za ostatak r vrijedi

$$r \in \mathbb{Z}_b := \{\, 0,\ 1,\ 2,\ \ldots\ ,\ b-1 \,\},$$

pa skup $\mathbb{Z}_b$ zovemo standardni sustav ostataka modulo b. ■

Zvuči poznato: ako uzmemo divizor $b = 2^n$, dobivamo skup $\mathbb{Z}_{2^n}$ svih prikazivih cijelih brojeva bez predznaka u računalu.

Dijeljenje cijelih brojeva — dvije operacije

Uočite da Euklidov teorem, odnosno, cjelobrojno dijeljenje s ostatkom daje dva rezultata:

- cjelobrojni kvocijent i ostatak.

Zgodno je odmah uvesti i oznake za obje operacije. Nažalost, nema standardne matematičke oznake za cjelobrojni kvocijent. Da izbjegnemo mogućnost bilo kakve zabune,

- za cjelobrojni kvocijent koristimo oznaku div, po ugledu na Pascal (oznaka / u C-u),

$$a \operatorname{div} b := q \in \mathbb{Z},$$

- a za ostatak postoji standardna oznaka mod, koju smo već koristili (oznaka % u C-u),

$$a \bmod b := r \in \mathbb{Z}_b.$$

Veza cjelobrojnog i običnog dijeljenja

Zahtjev da je $0 \leq r < b$, tj. da ostatak r pripada skupu $\mathbb{Z}_b$, daje jednostavnu vezu cjelobrojnog i običnog dijeljenja (onog u racionalnim brojevima).

Lako se vidi da vrijedi

$$a \text{ div } b = q = \left\lfloor \frac{a}{b} \right\rfloor .$$

Dakle, cjelobrojni kvocijent je "najveće cijelo" od običnog racionalnog kvocijenta.

Ova veza je specifična za izbor $r \in \mathbb{Z}_b$ i

- ne vrijedi za drugačije sustave ostataka,

a takve sustave možemo izabrati, kao što ćemo vidjeti.

Dijeljenje cijelih brojeva bez predznaka

Sve što smo dosad rekli o cjelobrojnom dijeljenju s ostatkom (zasad) vrijedi

- samo za cijele brojeve, ili, preciznije, na $\mathbb{Z} \times \mathbb{N}$.

Taj skup je "domena" za Euklidov teorem. Stoga su operacije div i mod definirane baš na toj domeni.

A što je s cjelobrojnim dijeljenjem s ostatkom u računalu, tj. na skupu $\mathbb{Z}_{2^n}$ prikazivih cijelih brojeva bez predznaka?

Odgovor: Potpuno isto kao da smo u cijelim brojevima.

Drugim riječima, dijeljenje s ostatkom cijelih brojeva bez predznaka je naprosto

- restrikcija "cjelobrojnih" operacija div i mod.

Cijeli brojevi bez predznaka — sažetak

Ako imamo n bitova za prikaz brojeva, onda je skup svih prikazivih cijelih brojeva bez predznaka jednak

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

Prikaz broja $B \in \mathbb{Z}_{2^n}$ dobiva se iz "proširenog" zapisa tog broja u bazi 2, s točno n binarnih znamenki.

Aritmetika cijelih brojeva bez predznaka je modularna aritmetika u prstenu $(\mathbb{Z}_{2^n}, \oplus_{2^n}, \odot_{2^n})$:

- operacije $+$, $-$ i $\cdot$ daju cjelobrojni rezultat modulo 2^n,
- operacije cjelobrojnog dijeljenja s ostatkom div i mod daju iste rezultate kao da dijelimo u $\mathbb{Z}$ (ili $\mathbb{N}_0$).

Cijeli brojevi u C-u — sažetak

U programskom jeziku C:

- cijelim brojevima bez predznaka odgovara tip koji se zove `unsigned int`, ili, skraćeno, `unsigned`,
- cijelim brojevima s predznakom odgovara tip koji se zove `int`.

Ovi tipovi postoje u nekoliko raznih veličina:

- standardna, `short`, `long`, a katkad i druge (`long long`).

Razlike su u broju bitova n predviđenih za prikaz.

- Zapis konstanti (vrijednost, navođenje tipa) — kasnije.
- Zapis operacija $+$, $-$ i $\cdot$ znakovima `+`, `-` i `*`.
- Zapis operacija div i mod znakovima `/` i `%`.

Cijeli brojevi bez predznaka u C-u — primjer 1

```
#include <stdio.h>

int main(void) {
    unsigned short i = 65535;  /* int ne pisem */

    printf("%d\n", i / 10);  /* 6553 */

    i = i + 3;
    printf("%d\n", i);       /* 2, a ne 65538 */

    return 0;
}
```

`USHRT_MAX = 65535` u zaglavlju `limits.h`. Ovdje je $n = 16$.

Cijeli brojevi bez predznaka u C-u — primjer 2

```
#include <stdio.h>

int main(void) {
    unsigned short i = 2, j = 4;

    i = i - j;
    printf("%d\n", i);  /* 65534, a ne -2 */

    return 0;
}
```

%d — piše vrijednost od i uz pretvorbu u tip int.
%hu — piše vrijednost od i baš kao unsigned short.

Ovdje nema razlike, ali da pišemo i - j, bilo bi razlike.

Cijeli brojevi u C-u — dodjeljivanje i čitanje

Napomena. Ista "modularna" priča vrijedi i

- kod dodjeljivanja i/ili čitanja vrijednosti cijelih brojeva!

Pretvorba iz niza znakova (dekadske znamenke u dekadskom zapisu broja) u binarni zapis (interni prikaz broja) ide

- onim aritmetičkim pravilima koja odgovaraju tipu broja, po "Hornerovom" algoritmu u bazi 10 (v. kasnije),
- a ta aritmetika je modularna.

Primjeri C programa:

- za dodjeljivanje i čitanje brojeva — malo kasnije,
- za prikaz brojeva — kasnije.

Prikaz cijelih brojeva s predznakom

Cijeli brojevi s predznakom

Cijeli brojevi s predznakom modeliraju skup $\mathbb{Z}$ cijelih brojeva.

Ako imamo n bitova na raspolaganju za prikaz, onda skup svih prikazivih brojeva ima (najviše) 2^n elemenata.

Među prikazivim brojevima moraju biti i (neki) negativni brojevi. Zgodno bi bilo da ih je podjednako mnogo kao i pozitivnih (odnosno, nenegativnih) brojeva.

Standardni dogovor: u računalu se prikazuje

- najveći mogući podskup uzastopnih brojeva iz $\mathbb{Z}$ koji je "skoro" simetričan oko 0.

Napomena. "Prava" simetrija oko nule dala bi neparan broj prikazivih brojeva — što nema smisla.

Cijeli brojevi s predznakom (nastavak)

Dakle, prikazivih negativnih brojeva ima podjednako mnogo kao i nenegativnih.

Ako želimo da točno polovina tih brojeva bude negativna, onda njih mora biti $2^n/2 = 2^{n-1}$. Nenegativnih brojeva je, naravno, isto toliko.

To znači da je skup svih prikazivih cijelih brojeva s predznakom jednak

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

Brojeve izvan tog skupa ne možemo prikazati (kao cijele brojeve s predznakom) koristeći samo n bitova.

Cijeli brojevi s predznakom (nastavak)

Najmanji i najveći prikazivi cijeli broj s predznakom su, redom:

$$-2^{n-1}, \qquad 2^{n-1} - 1.$$

Tipične vrijednosti za ta dva "granična" broja su:

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

Uočite da raspon prikazivih cijelih brojeva s predznakom nije jako velik, čak i za $n = 32$ (što je standard). **Oprez!**

Zato se danas sve više koristi $n = 64$ (želja za $n = 128$),

$$2^{63} - 1 \;=\; 9\,223\,372\,036\,854\,775\,807.$$

Prikaz cijelih brojeva s predznakom

Kako stvarno izgleda prikaz brojeva u tih n bitova?

Prikaz pojedinog (prikazivog) broja potpuno je određen s dva zahtjeva:

- nenegativni brojevi imaju isti prikaz kao u cijelim brojevima bez predznaka (dovoljno je to tražiti samo za jedan broj, na primjer, za nulu),
- aritmetika za te prikaze mora (kao i ranije, za brojeve bez predznaka) dati "dobru" algebarsku strukturu na cijelim brojevima s predznakom, tj. na skupu $\mathbb{Z}_{2^n}^-$.

Prvi zahtjev je dosta jasan, ali što znači drugi?

- Zatvorenost na operacije zbrajanja i množenja, dobra svojstva tih operacija!

Aritmetika cijelih brojeva s predznakom

Dakle, aritmetika za cijele brojeve s predznakom mora i dalje biti ista, tj.

- modularna aritmetika modulo 2^n,

samo je interpretacija sustava ostataka drugačija.

Skup svih prikazivih cijelih brojeva s predznakom

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

je, također, potpuni sustav ostataka modulo 2^n.

Na tom skupu opet možemo definirati operacije zbrajanja i množenja (bez posebnih oznaka) preko odgovarajućih ostataka cjelobrojnih operacija $+$ i $\cdot$. Kao i prije, dobivamo strukturu prstena s jedinicom, ali sad na skupu $\mathbb{Z}_{2^n}^{-}$.

Prikaz cijelih brojeva s predznakom (nastavak)

A sad je lako dobiti prikaz svih negativnih brojeva.

Usporedimo skupove prikazivih brojeva bez predznaka i s predznakom, tj. pripadne sustave ostataka modulo 2^n,

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

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

Pravilo za prikaz je vrlo jednostavno:

- iste prikaze imaju oni brojevi (iz ta dva skupa) koji imaju isti "pravi" ostatak modulo 2^n.

Zaključak: za $B = 1, \ldots, 2^{n-1}$,

- prikaz negativnog broja $-B$ (s predznakom)
- jednak je prikazu broja $2^n - B$ (bez predznaka). ■

Primjer — prikaz cijelih brojeva za $n = 3$

Primjer. Neka je $n = 3$, tj. imamo samo 3 bita za prikaz. Prikazivih brojeva ima $2^3 = 8$. Prikazi cijelih brojeva bez predznaka iz $\mathbb{Z}_8$ i cijelih brojeva s predznakom iz $\mathbb{Z}_8^-$ su:

$B \in \mathbb{Z}_8$	$B \in \mathbb{Z}_8^-$	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

Prikaz cijelih brojeva s predznakom — primjeri

Primjer. Prikaz broja -2^{n-1} dobivamo tako da

- uzmemo $B = 2^{n-1}$ (koji sam nije prikaziv s predznakom)
- i pogledamo prikaz broja $2^n - B$ bez predznaka:

$$-2^{n-1} \quad \longleftrightarrow \quad 2^n - 2^{n-1} = 2^{n-1} = [\,1\ 0\ 0\ \ldots\ 0\ 0\,].$$

Primjer. Broj -1 ima prikaz ($B = 1$)

$$-1 \quad \longleftrightarrow \quad 2^n - 1 = [\,1\ 1\ 1\ \ldots\ 1\ 1\,].$$

Prikaz cijelih brojeva s predznakom — primjeri

Primjer. Kad najvećem prikazivom broju s predznakom, a to je $2^{n-1}-1$, dodamo 1, rezultat je

- najmanji prikazivi broj s predznakom,

jer je

$$\begin{aligned}(2^{n-1}-1)+1 &= ((2^{n-1}-1)+1) \bmod 2^n \\ &= 2^{n-1} \bmod 2^n = -2^{n-1} \in \mathbb{Z}_{2^n}^-.\end{aligned}$$

Naravno, isti rezultat izlazi i običnim binarnim "zbrajanjem prikaza"

$$[\,0\;1\;1\;\ldots\;1\;1\,]+[\,0\;0\;0\;\ldots\;0\;1\,]=[\,1\;0\;0\;\ldots\;0\;0\,].$$

Uočite da nema prijenosa.

Prikaz cijelih brojeva s predznakom — primjeri

Primjer. Analogno, zbrajanju $(-1) + 1 = 0$ odgovara binarno "zbrajanje prikaza"

$$[\,1\;1\;1\;\ldots\;1\;1\,] + [\,0\;0\;0\;\ldots\;0\;1\,] = 1\,[\,0\;0\;0\;\ldots\;0\;0\,].$$

Ovdje imamo prijenos, ali se on "modularno" ignorira.

Prikaz cijelih brojeva s predznakom — komentar

Dakle, prikaz negativnih brojeva je posljedica

- one iste modularne aritmetike,

koja vrijedi i za cijele brojeve bez predznaka.

Tehnički gledano, tri osnovne operacije $+$, $-$ i $\cdot$ na cijelim brojevima, u računalu se izvršavaju

- potpuno jednako (istim "elektroničkim krugovima") za cijele brojeve s predznakom i bez njega,

Važni su samo bitovi u prikazu, a ne i njihova interpretacija!

Za cjelobrojno dijeljenje to ne vrijedi (v. malo kasnije).

Prikaz suprotnog broja preko komplementa

Za zadani broj $B > 0$, kad tražimo prikaz broja $-B$,

- broj $2^n - B$ nije baš lako izračunati "na ruke", čim je broj bitova n iole veći.

Srećom, ima i lakši način! Prikaz broja $-B$ možemo naći i tehnikom "dvojnog komplementa":

- prikaz zadanog broja B treba komplementirati, tj. pretvoriti nule u jedinice i obratno (tj. $0 \leftrightarrow 1$),
- a zatim, tom komplementu treba dodati 1 (modulo 2^n).

Kratko opravdanje: ako je $\bar{B}$ broj (bez predznaka) kojem odgovara komplementirani prikaz broja B, onda je (očito)

$$B + \bar{B} = [\ 1\ 1\ 1\ \ldots\ 1\ 1\ ] = 2^n - 1.$$

Prikaz suprotnog broja preko komplementa

Prebacimo 1 s desne na lijevu stranu

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

i zapišimo ovu relaciju "modularnim zbrajanjem" $\oplus_{2^n}$ u $\mathbb{Z}_{2^n}$

$$B \oplus_{2^n} (\bar{B} \oplus_{2^n} 1) = 0.$$

Dakle, jedinstveni suprotni element (inverz obzirom na zbrajanje) elementa $B \in \mathbb{Z}_{2^n}$ jednak je $\bar{B} \oplus_{2^n} 1$.

Drugim riječima, suprotni broj dobivamo tako da

- komplementiramo broj (odnosno, njegov prikaz) i
- dodamo 1 modulo 2^n.

Potpuno isto "pravilo" vrijedi i za negativne brojeve B!

Prikaz cijelih brojeva s predznakom — primjer

Primjer. Uzmimo da je $n = 8$ (budimo skromni) i pogledajmo zapis broja -120. Za početak, vrijedi

$$-2^7 = -128 \leq -120 \leq 127 = 2^7 - 1,$$

pa je -120 prikaziv. Nadalje, binarni prikaz broja $B = 120$ je

$$\begin{aligned} 120 &= 64 + 32 + 16 + 8 \\ &= 0 \cdot 2^7 + 1 \cdot 2^6 + 1 \cdot 2^5 + 1 \cdot 2^4 \\ &\quad + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0. \end{aligned}$$

Dakle, broj $B = 120$, kao cijeli broj bez predznaka, ima prikaz

$$120 \quad \longleftrightarrow \quad [\, 0\ 1\ 1\ 1\ 1\ 0\ 0\ 0 \,] \,.$$

Prikaz cijelih brojeva s predznakom — primjer

U zadanih $n = 8$ bitova, broj $B = 120$ (kao cijeli broj s predznakom, ili bez njega) ima prikaz

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

Komplementiranjem prikaza (bit po bit) dobivamo

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

Dodamo 1 (modulo 2^8). Prikaz broja $-B = -120$ je

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

Prikaz cijelih brojeva s predznakom — primjer

Istu stvar možemo dobiti i tako da nađemo prikaz broja

$$2^8 - B = 256 - 120 = 136,$$

kao cijelog broja bez predznaka. Imamo redom

$$\begin{aligned} 136 &= 128 + 8 \\ &= 1 \cdot 2^7 + 0 \cdot 2^6 + 0 \cdot 2^5 + 0 \cdot 2^4 \\ &\quad + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0. \end{aligned}$$

Za $n = 8$, broj -120, kao cijeli broj s predznakom, ima isti prikaz kao i 136, bez predznaka

$$-120 \quad \longleftrightarrow \quad 136 \quad \longleftrightarrow \quad [\,1\;0\;0\;0\;1\;0\;0\;0\,]\,.$$

Dijeljenje cijelih brojeva s predznakom

Dijeljenje cijelih brojeva — izbor ostatka

Standardno ograničenje na ostatak $0 \leq r < b$, tj. $r \in \mathbb{Z}_b$, prirodno odgovara cijelim brojevima bez predznaka.

Zato, u većini programskih jezika (uključivo i C) vrijedi da

- stvar radi očekivano, tj. prema standardnom Euklidovom teoremu, samo na skupu $\mathbb{N}_0 \times \mathbb{N}$.

Dakle, za nenegativne brojeve s predznakom dobivamo očekivane (i korektne) rezultate u cjelobrojnom dijeljenju.

Međutim, kod brojeva s predznakom imamo i negativne brojeve, pa (možda) ima smisla dozvoliti

- da i ostaci budu negativni, u nekim slučajevima.

Odluka, tj. izbor ostataka — ovisi o primjeni i željenim svojstvima rezultata!

Dijeljenje cijelih brojeva — izbor ostatka

Pripadni "Euklidov teorem" smije imati i kompliciraniju formulaciju, ako je to korisno u praksi.

- Pitanje je samo je li izbor ostataka propisan ili ne!!!

Nažalost, za negativne operande, ponašanje dijeljenja

- ne mora biti precizno definirano!

Na primjer, stari C90 standard (knjiga KR2) kaže:

- ako je barem jedan od dva operanda negativan, rezultat ovisi o implementaciji.

Dakle, nije predvidiv — isti program može davati različite rezultate, ovisno o računalu i izboru C compilera.
Zato — čitajte upute ili, naprosto, probajte!

Srećom, novi C99 standard precizno propisuje izbor ostataka.

Dijeljenje cijelih brojeva s predznakom

Eksperiment:

- test–program `divmod.c` (v. sljedeća stranica),
- Intel C/C++ compiler, `gcc` compiler (Code::Blocks).

Rezultati $q = a \text{ div } b$ i $r = a \bmod b$ za $a = \pm 5$, $b = \pm 3$:

a	b	q	r
5	3	1	2
-5	3	-1	-2
5	-3	-1	2
-5	-3	1	-2

Operacije div i mod interpretiramo na $\mathbb{Z} \times (\mathbb{Z} \setminus \{0\})$.

Test–program `divmod.c`

```
#include <stdio.h>

int main(void) {
    char s[] =
        " a = %2d, b = %2d, q = %2d, r = %2d\n";
    int a, b;

    a =  5; b =  3;  printf(s, a, b, a/b, a%b);
    a = -5; b =  3;  printf(s, a, b, a/b, a%b);
    a =  5; b = -3;  printf(s, a, b, a/b, a%b);
    a = -5; b = -3;  printf(s, a, b, a/b, a%b);

    return 0;
}
```

Veza cjelobrojnog i običnog dijeljenja (standard)

Ključ za interpretaciju:

- kvocijent se uvijek "zaokružuje" prema nuli,

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

- ostatak ima isti predznak kao i a.

$$r = \operatorname{sign}(a) \cdot (|a| \bmod |b|).$$

Za ostatak r ovdje vrijedi:

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

Ovo je modificirani "Euklidov teorem" za cijele brojeve u C-u!

Dijeljenje cijelih brojeva (standard)

Novi C99 standard propisuje ovakav izbor ostataka, tj.

- ovakvo ponašanje cjelobrojnog dijeljenja za cijele brojeve s predznakom.

Zato zaboravite raniju definiciju ("pravi" Euklidov teorem), iako se "novi" mod ponaša drugačije nego u matematici.

Prednosti ovakve definicije operacija div i mod na skupu $\mathbb{Z} \times (\mathbb{Z} \setminus \{0\})$:

- bez obzira na predznake od a i b, dobivamo
- iste apsolutne vrijednosti kvocijenta q i ostatka r,

tj. samo predznaci od q i r ovise o predznacima od a i b.

Ovo je i najčešća realizacija cjelobrojnog dijeljenja u praksi (misli se i na ostale programske jezike).

Cijeli brojevi s predznakom — sažetak

Ako imamo n bitova za prikaz brojeva, onda je skup svih prikazivih cijelih brojeva s predznakom jednak

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

Za prikaz broja $B \in \mathbb{Z}_{2^n}^{-}$ vrijedi:

- nenegativni brojevi $B = 0, \ldots, 2^{n-1}-1$ imaju isti prikaz kao i bez predznaka,
- negativni brojevi $B = -1, \ldots, -2^{n-1}$ imaju isti prikaz kao i brojevi $2^n + B$ bez predznaka.

Cijeli brojevi s predznakom — sažetak

Osim toga, prikaz suprotnog broja $-B$ dobivamo tako da

- komplementiramo prikaz samog broja B i
- dodamo 1 modulo 2^n.

Aritmetika cijelih brojeva s predznakom je modularna aritmetika modulo 2^n na sustavu ostataka $\mathbb{Z}_{2^n}^-$.

- To vrijedi za operacije $+$, $-$ i $\cdot$.

Operacije cjelobrojnog dijeljenja s ostatkom div i mod daju iste rezultate kao da dijelimo u $\mathbb{Z}$ (= "neki" Euklidov teorem).

- Za svaki slučaj, treba provjeriti kako se dobiva proširenje ovih operacija s $\mathbb{N}_0 \times \mathbb{N}$ na $\mathbb{Z} \times (\mathbb{Z} \setminus \{0\})$,
- tj. radi li compiler prema C99 standardu.

Cijeli brojevi u C-u — sažetak

U programskom jeziku C:

- cijelim brojevima s predznakom odgovara tip koji se zove `int`.

Ovaj tip postoji u nekoliko raznih veličina, a razlike su u broju bitova n predviđenih za prikaz.

Na 32-bitnim arhitekturama računala imamo sljedeće tipove:

- standardni `int` ($n = 32$),
- `short` ($n = 16$),
- `long` ($n = 32$), tj. isto kao standardni `int`,
- a katkad i druge, poput `long long` ($n = 64$).

Zapis konstanti (vrijednost, navođenje tipa) — kasnije.

Cijeli brojevi s predznakom u C-u — primjer 1

```
#include <stdio.h>

int main(void) {
    short int i = 32766;  /* n = 16 za short */

    i += 1;
    printf("%d\n", i);  /* 32767 */
    i += 1;
    printf("%d\n", i);  /* -32768, a ne 32768 */

    return 0;
}
```

`SHRT_MAX = 32767` u zaglavlju `limits.h`. Ovdje je $n = 16$.

Cijeli brojevi — dodjeljivanje

Primjer. Modularna aritmetika kod dodjeljivanja.

```
#include <stdio.h>

int main(void) {
    int broj;

    broj = 5;           printf(" broj = %d\n", broj);
    broj = 2000000000;  printf(" broj = %d\n", broj);
    broj = 4000000000;  printf(" broj = %d\n", broj);
    broj = 8000000000;  printf(" broj = %d\n", broj);

    return 0;
}
```

Cijeli brojevi — dodjeljivanje (nastavak)

U tipu `int`, broj bitova za prikaz brojeva je $n = 32$.

Izlaz programa je:

```
broj = 5
broj = 2000000000
broj = -294967296
broj = -589934592
```

Cijeli brojevi — čitanje

Primjer. Modularna aritmetika kod čitanja.

```
#include <stdio.h>

int main(void) {
    int broj;

    scanf("%d", &broj);
    printf(" ucitani broj = %d\n", broj);

    return 0;
}
```

Za ulaz 4000000000,
izlaz programa je: `ucitani broj = -294967296`.

Aritmetika cijelih brojeva: klasične greške

Cijeli brojevi — klasične greške

Primjer. Računanje $n!$ u cjelobrojnoj aritmetici.

Za prirodni broj $n \in \mathbb{N}$, funkciju faktorijela definiramo na sljedeći način:

$$1! = 1,$$

$$n! = n \cdot (n-1)!, \quad \text{za} \quad n \geq 2.$$

Napišimo C program koji računa broj $50!$ u cjelobrojnoj aritmetici (tip `int`).

Očito je $n!$ = produkt svih prirodnih brojeva od 1 do n,

$$n! = 1 \cdot 2 \cdots (n-1) \cdot n,$$

pa koristimo množenje u petlji, a ne rekurziju.

Cijeli brojevi — klasične greške (nastavak)

```
#include <stdio.h>

int main(void) {
    int i, f50 = 1;  /* n = 32 za int */

    for (i = 2; i <= 50; ++i)
        f50 *= i;

    printf(" f50 = %d\n", f50);  /* f50 = 0 */

    return 0;
}
```

Izlaz programa je: `f50 = 0`. Zašto?

Cijeli brojevi — klasične greške (nastavak)

Za početak, 50! je ogroman broj. Točna vrijednost je

$$50! = 30414\ 09320\ 17133\ 78043\ 61260\ 81660$$
$$64768\ 84437\ 76415\ 68960\ 51200\ 00000\ 00000,$$

i ima 65 dekadskih znamenki. Dakle, sigurno nije prikaziv u cjelobrojnoj aritmetici.

Granice za tip `int` ($n = 32$ bita) iz zaglavlja `limits.h` su

$$\texttt{INT\_MAX} = 2147483647, \quad \texttt{INT\_MIN} = (-\texttt{INT\_MAX} - 1).$$

Objasnimo još zašto je `f50 = 0` u našem programu. Cjelobrojna aritmetika u kojoj računamo je modularna aritmetika — modulo 2^{32}.

Cijeli brojevi — klasične greške (nastavak)

To znači da naš program kaže da je

$$50! = 0 \bmod 2^{32},$$

ili da 2^{32} dijeli $50!$.

Zadatak. Nađite najveću potenciju broja 2 koja dijeli $50!$, ili, općenito, $n!$. U traženoj potenciji 2^m, eksponent m je

$$m = \left\lfloor \frac{n}{2} \right\rfloor + \left\lfloor \frac{n}{2^2} \right\rfloor + \left\lfloor \frac{n}{2^3} \right\rfloor + \left\lfloor \frac{n}{2^4} \right\rfloor + \left\lfloor \frac{n}{2^5} \right\rfloor + \cdots .$$

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

Zadatak. Nađite najmanji prirodni broj n za koji je $n! = 0$, u cjelobrojnoj aritmetici s ℓ bitova za prikaz brojeva ($\bmod 2^\ell$).

Tablica $n!$ za $\ell = 16$ i $\ell = 32$ bita

Usporedimo rezultate za $n!$ dobivene u cjelobrojnoj aritmetici s $\ell = 16$ bitova (tip `short int`) i $\ell = 32$ bita (tip `int`), s (ispravnim) rezultatom dobivenim u realnoj aritmetici.

$n!$	$\ell = 16$	$\ell = 32$	realna aritmetika
7!	5040	5040	5040
8!	−25216	40320	40320
9!	−30336	362880	362880
10!	24320	3628800	3628800
11!	5376	39916800	39916800
12!	−1024	479001600	479001600

Tablica $n!$ za $\ell = 16$ i $\ell = 32$ bita (nastavak)

$n!$	$\ell = 16$	$\ell = 32$	realna aritmetika
13!	−13312	1932053504	6227020800
14!	10240	1278945280	87178291200
15!	22528	2004310016	1307674368000
16!	−32768	2004189184	20922789888000
17!	−32768	−288522240	355687428096000
18!	0	−898433024	6402373705728000
19!	0	109641728	121645100408832000
20!	0	−2102132736	2432902008176640000

Dodatak — o prikazu cijelih brojeva

Sadržaj

- Cijeli brojevi — prikaz i aritmetika:
  - Prikaz cijelih brojeva bez predznaka (precizno).
  - Modularna aritmetika, aritmetika adresa (primjer).
  - Modularna aritmetika, prsten ostataka modulo 2^n.
  - Dijeljenje cijelih brojeva — Euklidov teorem, cjelobrojni kvocijent i ostatak.
  - Dijeljenje s ostatkom u prstenu $\mathbb{Z}_b$.
  - Prikaz brojeva s predznakom — sustav ostataka.
  - Vodeći bit i predznak.
  - Prikaz suprotnog broja — precizna argumentacija za "komplementiraj i dodaj 1".
  - Dijeljenje cijelih brojeva s predznakom — izbor ostataka.

Prikaz brojeva bez predznaka (precizno)

Prikaz cijelih brojeva bez predznaka (precizno)

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

Ako je $B = 0$, onda je svih n bitova u prikazu jednako 0, tj.

$$B = 0 \iff \text{bit}_i = 0, \quad \text{za} \quad i = 0, \ldots, n-1.$$

U protivnom, ako je $B > 0$, onda njegov normalizirani pozicioni prikaz u bazi 2 ima oblik:

$$B = b_k \cdot 2^k + \cdots + b_1 \cdot 2 + b_0, \quad b_i \in \{0, 1\}, \quad b_k > 0,$$

gdje su b_i binarne znamenke (bitovi) broja B.

Ograničenje $b_k > 0$ na vodeću binarnu znamenku samo kaže da je prikaz normaliziran, tj. da nema nul–znamenki "sprijeda".

Prikaz cijelih brojeva bez predznaka (precizno)

Nadalje, znamo da je $k \leq n-1$, jer je B prikaziv.

Ako pišemo samo binarne znamenke (bitove) "u nizu", pripadni pozicioni zapis broja B u bazi 2 ima oblik

$$B = (b_k \; b_{k-1} \; \cdots \; b_1 \; b_0)_2.$$

I točno tako se spremaju bitovi u prikazu, samo treba vodeće bitove dopuniti nulama, od indeksa $k+1$ do $n-1$, ako takvih ima, tj. ako je $k < n-1$.

Dakle, prikaz broja B kao cijelog broja bez predznaka ima oblik

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

Prikaz cijelih brojeva bez predznaka (precizno)

U računalu će to biti prikazano kao "cjelina" od n bitova

$$\text{bit}_{n-1}\ \text{bit}_{n-2}\ \ldots\ \text{bit}_1\ \text{bit}_0.$$

Ako "proširimo" zapis broja B u bazi 2 do točno n binarnih znamenki,

$$B = b_{n-1} \cdot 2^{n-1} + \cdots + b_1 \cdot 2 + b_0, \quad b_i \in \{0, 1\},$$

s tim da vodeće znamenke smiju biti 0, odmah dobivamo prikaz broja B kao cijelog broja bez predznaka

$$\text{bit}_i = b_i, \quad \text{za} \quad i = 0, \ldots, n-1,$$

s tim da ovo vrijedi i za $B = 0$.

Aritmetika adresa — primjer u Windows XP

Nažalost, u praksi postoje razne "čarolije" na temu adresa, koje je katkad zgodno znati.

Primjer. U neko doba, 32-bitni MS Windows XP imao je "ograničenje" adresa na 2 GB, što je polovina od normalnog adresnog prostora na IA–32 (4 GB).

- Stvar je išla tako daleko da se XP "nije dizao", ako u računalu imate više od 2 GB memorije (probao sam).
- Sa zadnjim SP3, uz malo truda, može ga se instalirati, ali se instalacija prilično "čudno" ponaša.

Ako naknadno dodate memoriju (onu preko 2 GB), onda i dalje "ne vidi" više od 3.25 GB!

Prsten ostataka modulo 2^n

Naime, skup svih prikazivih cijelih brojeva bez predznaka

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

je ujedno i standardni sustav ostataka koji dobivamo pri cjelobrojnom dijeljenju s 2^n. Zato se i označava sa $\mathbb{Z}_{2^n}$.

Ako na njemu definiramo binarne operacije zbrajanja $\oplus$ i množenja $\odot$ preko ostataka cjelobrojnih operacija $+$ i $\cdot$,

$$A \oplus B := (A + B) \bmod 2^n,$$
$$A \odot B := (A \cdot B) \bmod 2^n,$$

onda $(\mathbb{Z}_{2^n}, \oplus, \odot)$ ima algebarsku strukturu prstena s 1.

U algebri se operacije $\oplus$ i $\odot$ obično označavaju s $\oplus_{2^n}$ i $\odot_{2^n}$.

Prsten ostataka modulo 2^n (nastavak)

Što znači da je $(\mathbb{Z}_{2^n}, \oplus, \odot)$ prsten s jedinicom? Vrijedi:

- $(\mathbb{Z}_{2^n}, \oplus)$ je komutativna grupa (obzirom na zbrajanje),
- $(\mathbb{Z}_{2^n}, \odot)$ je polugrupa (obzirom na množenje), čak i monoid, jer ima jedinicu $1 \in \mathbb{Z}_{2^n}$,
- operacije $\oplus$ i $\odot$ vezane su zakonom distributivnosti, tj.

$$A \odot (B \oplus C) = A \odot B \oplus A \odot C,$$
$$(A \oplus B) \odot C = A \odot C \oplus B \odot C,$$

za svaki izbor A, B, $C \in \mathbb{Z}_{2^n}$.

Dodatno, $(\mathbb{Z}_{2^n}, \oplus, \odot)$ je i komutativni prsten s jedinicom, ali nije polje (za $n > 1$), jer ima djelitelja nule ($2 \cdot 2^{n-1} = 0$).

Prsten ostataka modulo 2^n (nastavak)

Neka je "$-A$" jedinstveni suprotni element elementa A obzirom na zbrajanje. Očito je "-0" $= 0$, a za $A \neq 0$ vrijedi "$-A$" $= 2^n - A$, jer je

$$A \oplus \text{“}-A\text{”} = (A + (2^n - A)) \bmod 2^n = 0.$$

Na kraju, oduzimanje $\ominus$ definiramo kao zbrajanje sa suprotnim elementom

$$A \ominus B := A + \text{“}-B\text{”}.$$

I tako smo dobili tri osnovne aritmetičke operacije na $\mathbb{Z}_{2^n}$. One se upravo na taj način realiziraju u računalu, za cijele brojeve bez predznaka, s n bitova za prikaz.

U programskim jezicima se ove tri operacije pišu znakovima `+`, `-` i `*` (za $\cdot$, odnosno $\odot$).

Dijeljenje s ostatkom (precizno)

Dijeljenje cijelih brojeva

A što je s dijeljenjem? To dosad nismo ni spomenuli!

S razlogom! "Obično" dijeljenje ima smisla tek u strukturi polja, poput racionalnih brojeva $\mathbb{Q}$, ili realnih brojeva $\mathbb{R}$.

Znamo da $\mathbb{N}_0$ i $\mathbb{Z}$ nisu polja. Isto vrijedi i za $\mathbb{Z}_{2^n}$, čim je $n > 1$, a slučaj $n = 1$ je potpuno neinteresantan za praksu, barem što se tiče aritmetike cijelih brojeva (iako je vrlo bitan za logičku algebru).

Što sad?

Zamjena za "obično" dijeljenje u cijelim brojevima je tzv. dijeljenje s ostatkom. Podloga za to je poznati Euklidov teorem o dijeljenju s ostatkom u skupu $\mathbb{Z}$.

Dijeljenje cijelih brojeva — Euklidov teorem

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

$$a = q \cdot b + r \quad \text{i} \quad 0 \leq r < b.$$

Broj q je cjelobrojni kvocijent, a r ostatak pri dijeljenju a s b.

Ograničenje $0 \leq r < b$ znači da za ostatak r vrijedi

$$r \in \mathbb{Z}_b := \{\, 0,\ 1,\ 2,\ \ldots\ ,\ b-1 \,\},$$

pa skup $\mathbb{Z}_b$ zovemo standardni sustav ostataka modulo b. ■

Zvuči poznato: ako uzmemo divizor $b = 2^n$, dobivamo skup $\mathbb{Z}_{2^n}$ svih prikazivih cijelih brojeva bez predznaka u računalu.

Dijeljenje cijelih brojeva — dvije operacije

Uočite da Euklidov teorem, odnosno, cjelobrojno dijeljenje s ostatkom daje dva rezultata:

- cjelobrojni kvocijent i ostatak.

Zgodno je odmah uvesti i oznake za obje ove operacije.

Nažalost, nema standardne matematičke oznake za

- cjelobrojni kvocijent.

Oznaka / standardno se koristi za operaciju "običnog" dijeljenja u poljima, ili se pišu razlomci. Kad napišem

$$a/b \qquad \text{ili} \qquad \frac{a}{b}$$

to odmah asocira na obično dijeljenje (što nije zgodno).

Oznake za cjelobrojni kvocijent i ostatak

S druge strane, u nekim programskim jezicima (C, Fortran) oznaka `/` se koristi i za cjelobrojno dijeljenje, ali to vrijedi

- ako i samo ako su oba operanda cijeli brojevi.

Usput, isti princip da tip rezultata ovisi o tipu oba operanda (tzv. "operator overloading") vrijedi i za tri ranije operacije s oznakama `+`, `-`, `*`.

Da izbjegnemo mogućnost bilo kakve zabune,

- za cjelobrojni kvocijent koristimo oznaku div, po ugledu na Pascal (oznaka `/` u C-u),
- a za ostatak postoji standardna oznaka mod, koju smo već koristili (oznaka `%` u C-u).

Definicija operacija za cjelobrojno dijeljenje

Precizna definicija ovih operacija izlazi direktno iz Euklidovog teorema.

Definicija. Neka su $a \in \mathbb{Z}$ i $b \in \mathbb{N}$ bilo koji brojevi, i neka su $q \in \mathbb{Z}$ (cjelobrojni kvocijent) i $r \in \mathbb{Z}_b$ (ostatak) jedinstveni brojevi za koje vrijedi

$$a = q \cdot b + r.$$

Operacije div i mod definiramo relacijama

$$a \text{ div } b := q \in \mathbb{Z}, \quad a \text{ mod } b := r \in \mathbb{Z}_b.$$

Ova definicija operacije mod točno odgovara onom što smo ranije koristili.

Dijeljenje cijelih brojeva — mala digresija

Za početak, uočite da su obje operacije definirane na skupu $\mathbb{Z} \times \mathbb{N}$, a kodomene su im različite.

Možda nekog zanima što se zbiva na skupu $\mathbb{Z} \times (\mathbb{Z} \setminus \{0\})$,

- kad cjelobrojno dijelimo dva cijela broja (što, naravno, ima smisla).

O tome malo kasnije, kod cijelih brojeva s predznakom.
Odmah jedno upozorenje:

- stvar radi očekivano (prema Euklidovom teoremu) samo na skupu $\mathbb{N}_0 \times \mathbb{N}$.

Trenutno nam upravo taj skup i treba, da dobijemo cjelobrojno dijeljenje za cijele brojeve bez predznaka (koji modeliraju skup $\mathbb{N}_0$).

Prsten ostataka modulo b (dodatak)

Za bilo koji fiksni divizor $b \geq 2$, na standardnom sustavu ostataka modulo b, tj. skupu

$$\mathbb{Z}_b = \{\, 0,\ 1,\ 2,\ \ldots\ ,\ b-1 \,\}$$

možemo definirati binarne operacije zbrajanja $\oplus_b$ i množenja $\odot_b$, preko ostataka cjelobrojnih operacija $+$ i $\cdot$,

$$A \oplus_b B := (A + B) \bmod b,$$

$$A \odot_b B := (A \cdot B) \bmod b.$$

Lako se dokazuje da $(\mathbb{Z}_b, \oplus_b, \odot_b)$ ima algebarsku strukturu komutativnog prstena s jedinicom (kao i ranije za $b = 2^n$).

Ta struktura je polje, ako i samo ako je b prost broj.

Dijeljenje s ostatkom u $\mathbb{Z}_b$ (dodatak)

Zašto su rezultati isti? Uzmimo bilo koji fiksni divizor $b \geq 2$.

Neka su $A \in \mathbb{Z}_b$ i $B \in \mathbb{Z}_b$, uz $B > 0$, bilo koji brojevi iz $\mathbb{Z}_b$ koje "smijemo dijeliti". Podijelimo ih cjelobrojno, i pokažimo da kvocijent $Q := A \operatorname{div} B$ i ostatak $R := A \bmod B$ moraju, također, pripadati skupu $\mathbb{Z}_b$.

Znamo da općenito vrijedi $Q \in \mathbb{Z}$, $R \in \mathbb{Z}_B$ (tj. $0 \leq R < B$) i

$$A = Q \cdot B + R.$$

Zbog $0 \leq A, B < b$, odmah vidimo da je

$$0 \leq Q < b \quad \text{i} \quad 0 \leq R < B < b,$$

što dokazuje Q, $R \in \mathbb{Z}_b$. Naravno, to vrijedi i za $b = 2^n$. ■

Dijeljenje cijelih brojeva bez predznaka

Zbog toga, dijeljenje s ostatkom u skupu $\mathbb{Z}_{2^n}$ prikazivih cijelih brojeva bez predznaka u računalu

- daje potpuno iste rezultate kao da dijelimo u $\mathbb{Z}$ (ili $\mathbb{N}_0$).

Dakle, nema ostataka modulo 2^n i "čarolija" s prikazivošću rezultata (nije potrebno, jer radi i bez toga).

Operacije div i mod su jedine aritmetičke operacije koje na prikazivim cijelim brojevima bez predznaka $\mathbb{Z}_{2^n}$ daju iste rezultate kao i na skupu $\mathbb{N}_0$ kojeg modeliramo.

Ponovimo još jednom da ostale tri operacije $+$, $-$ i $\cdot$

- daju cjelobrojni rezultat modulo 2^n.

Euklidov teorem u prstenu $\mathbb{Z}_b$ (dodatak)

Uočite da, za operacije div i mod na skupu $\mathbb{Z}_b$, koristimo Euklidov teorem za cijele brojeve, tj. na domeni $\mathbb{Z} \times \mathbb{N}$.

Pitanje: Kad znamo da je $(\mathbb{Z}_b, \oplus_b, \odot_b)$ prsten, kao i $(\mathbb{Z}, +, \cdot)$, zašto ne uzmemo "prirodniju" domenu $\mathbb{Z}_b \times (\mathbb{Z}_b \setminus \{0\})$?

Odgovor: Na toj domeni, s pripadnim modularnim operacijama $\oplus_b$ i $\odot_b$, također, vrijedi Euklidov teorem, ali nema jedinstvenosti rezultata.

Euklidov teorem u prstenu $(\mathbb{Z}_b, \oplus_b, \odot_b)$. Za bilo koja dva broja A, $B \in \mathbb{Z}_b$, uz $B > 0$, postoje brojevi Q, $R \in \mathbb{Z}_b$, takvi da je

$$A = Q \odot_b B \oplus_b R \quad \text{i} \quad 0 \leq R < B,$$

ali ti brojevi ne moraju biti jedinstveni. ■

Euklidov teorem u prstenu $\mathbb{Z}_b$ (dodatak)

Primjer. Uzmimo $b = 2^3 = 8$, tj. prsten $(\mathbb{Z}_8, \oplus_8, \odot_8)$, i brojeve $A = 5$, $B = 4$. Onda je, kao u cijelim brojevima,

$$5 = 1 \cdot 4 + 1,$$

tj. $5 \text{ div } 4 = 1$ i $5 \bmod 4 = 1$. Ali, zbog $2 \odot_8 4 = 0 \bmod 8$, vrijedi i

$$5 = 3 \odot_8 4 \oplus_8 1 = 13 \bmod 8,$$

$$5 = 5 \odot_8 4 \oplus_8 1 = 21 \bmod 8,$$

$$5 = 7 \odot_8 4 \oplus_8 1 = 29 \bmod 8.$$

Modularni kvocijenti (s ostatkom 1) su 1, 3, 5 i 7, a najmanji je "pravi".

Ostatak, također, nije jedinstven, jer je $5 = 2 \odot_8 4 \oplus_8 5$. ■

Prikaz cijelih brojeva s predznakom (precizno)

Veza između prikaza i aritmetike

Kod brojeva bez predznaka, prikaz binarnim znamenkama je očito odgovarao aritmetici — recimo, ovako:

- kad prikazu broja B (kao nizu bitova u bazi 2) dodamo 1, dobijemo točno prikaz broja $B + 1$.

(Hm, nismo baš puno razmišljali o tome).

Modularna aritmetika modulo 2^n dodatno još "zatvara krug" na n bitova, tj. daje zatvorenost operacija, a onda i dobru strukturu prstena s jedinicom na $\mathbb{Z}_{2^n} = \{\, 0,\ 1,\ \ldots\ ,\ 2^n - 1 \,\}$.

Zato gore, umjesto običnog cjelobrojnog $+$, koristimo $\oplus_{2^n}$.

Potpuno isto mora vrijediti i za prikaze cijelih brojeva s predznakom iz skupa $\mathbb{Z}^-_{2^n}$ — inače nemamo jednostavnu realizaciju aritmetike i dobru strukturu.

Prikaz cijelih brojeva s predznakom (dodatak)

Detaljnija argumentacija ovog zaključka ide na sljedeći način.

Prvo "poklopimo" zajednički dio nenegativnih brojeva

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

u ta dva skupa. Svaki od tih brojeva ima isti prikaz u oba skupa, pa ih "sparujemo"

$$B \in \mathbb{Z}^{-}_{2^n} \quad \longleftrightarrow \quad B \in \mathbb{Z}_{2^n},$$

za $B = 0, \ldots, 2^{n-1}-1$.

Uočite da

- prikazi svih ovih brojeva imaju vodeći bit jednak 0.

Prikaz cijelih brojeva s predznakom (dodatak)

Zatim "zatvaramo modularni krug", dodavanjem jedan po jedan modulo 2^n, i "sparujemo" odgovarajuće brojeve.

Prvi "dodaj jedan" modulo 2^n daje

$$((2^{n-1}-1)+1) \bmod 2^n = 2^{n-1} \bmod 2^n = \begin{cases} 2^{n-1} \in \mathbb{Z}_{2^n}, \\ -2^{n-1} \in \mathbb{Z}^-_{2^n}, \end{cases}$$

tj. zatvara krug u $\mathbb{Z}^-_{2^n}$, pa "sparujemo"

$$-2^{n-1} \in \mathbb{Z}^-_{2^n} \quad \longleftrightarrow \quad 2^{n-1} \in \mathbb{Z}_{2^n}.$$

A dalje sve ide redom.

Prikaz cijelih brojeva s predznakom (dodatak)

Kad "dodaj jedan" ponovimo k puta, tj. kad broju $2^{n-1}-1$ dodamo k modulo 2^n, izlazi:

$$-2^{n-1}+(k-1) \in \mathbb{Z}_{2^n}^- \quad \longleftrightarrow \quad (2^{n-1}-1)+k \in \mathbb{Z}_{2^n},$$

za $k = 1, \ldots, 2^{n-1}$. Supstitucijom $-B = -2^{n-1}+(k-1)$, za negativne brojeve dobivamo

$$-B \in \mathbb{Z}_{2^n}^- \quad \longleftrightarrow \quad 2^n - B \in \mathbb{Z}_{2^n},$$

i to vrijedi za $B = 1, \ldots, 2^{n-1}$.

Dakle, spareni brojevi razlikuju se za točno 2^n.

To je jasno, jer moraju imati isti "pravi" ostatak modulo 2^n.

Prikaz cijelih brojeva s predznakom (dodatak)

Zaključak: za $B = 1, \ldots, 2^{n-1}$,

- prikaz negativnog broja $-B$ (s predznakom)
- jednak je prikazu broja $2^n - B$ (bez predznaka).

Uočite da

- prikazi svih ovih brojeva imaju vodeći bit jednak 1. ■

Vodeći bit — predznak

U prikazu cijelih brojeva s predznakom

- svi nenegativni brojevi imaju vodeći bit jednak 0, a
- svi negativni brojevi imaju vodeći bit jednak 1.

Zato se vodeći bit, obično, zove i bit predznaka (engl. sign).

Taj naziv, nažalost, može zavesti na pogrešnu ideju o prikazu. Prikaz cijelih brojeva s predznakom ne dobiva se kao:

- bit predznaka i prikaz apsolutne vrijednosti broja (bez predznaka, s bitom manje).

Za preciznu argumentaciju — v. dodatak (sljedeći odjeljak).

Za razliku od ovog, kod "realnih" brojeva to vrijedi, što samo povećava mogućnost zabune.

Komplementiraj i dodaj 1 (precizna argumentacija)

Veza broja i prikaza s predznakom

Zasad znamo da za prikaz cijelih brojeva s predznakom vrijedi:

- nenegativni brojevi $B = 0, \ldots, 2^{n-1} - 1$ imaju isti prikaz kao i bez predznaka,
- negativni brojevi $B = -1, \ldots, -2^{n-1}$ imaju isti prikaz kao i brojevi $2^n + B$ bez predznaka.

Tj., imamo dva različita slučaja, ovisno o "predznaku" broja.

Da bismo dobili vezu između n bitova u prikazu broja B s predznakom i samog broja, postupamo na sljedeći način.

Broju $B \in \mathbb{Z}_{2^n}^-$ pridružimo broj $B' \in \mathbb{Z}_{2^n}$, tako da

- B' bez predznaka ima isti prikaz na n bitova kao i broj B s predznakom.

Taj broj B' označavamo s prikaz(B).

Veza broja i prikaza s predznakom (nastavak)

Već znamo da je

$$\text{prikaz}(B) = \begin{cases} B, & \text{za} \quad B = 0, \ldots, 2^{n-1} - 1, \\ 2^n + B, & \text{za} \quad B = -1, \ldots, -2^{n-1}. \end{cases}$$

Nama treba "obratna" veza:

$$B = \begin{cases} \text{prikaz}(B), & \text{za} \quad \text{prikaz}(B) = 0, \ldots, 2^{n-1} - 1, \\ \text{prikaz}(B) - 2^n, & \text{za} \quad \text{prikaz}(B) = 2^{n-1}, \ldots, 2^n - 1, \end{cases}$$

jer $\text{prikaz}(B)$ ima jednostavnu vezu sa svojim bitovima.

A onda je lako.

Veza broja i prikaza s predznakom (nastavak)

Neka je

$$[\,\mathrm{bit}_{n-1}\ \mathrm{bit}_{n-2}\ \ldots\ \mathrm{bit}_1\ \mathrm{bit}_0\,]$$

zapis prikazivog cijelog broja $B \in \mathbb{Z}^-_{2^n}$ s predznakom, gdje su $\mathrm{bit}_i \in \{0,1\}$ bitovi u prikazu, za $i = 0, \ldots, n-1$.

Po definiciji, $\mathrm{prikaz}(B)$ ima isti zapis bez predznaka, pa su bitovi bit_i baš binarne znamenke broja $\mathrm{prikaz}(B)$, tj. vrijedi

$$\mathrm{prikaz}(B) := \mathrm{bit}_{n-1} \cdot 2^{n-1} + \cdots + \mathrm{bit}_1 \cdot 2 + \mathrm{bit}_0.$$

Supstitucijom u "obratnu" vezu odmah dobivamo relacije za broj B preko bitova u njegovom prikazu s predznakom.

Veza broja i prikaza s predznakom (nastavak)

Opet imamo dva slučaja, ovisno o "predznaku":

- $B = 0, \ldots, 2^{n-1} - 1$, tj. $B \geq 0 \iff \text{bit}_{n-1} = 0$, i tada je

$$\begin{aligned} B &= 0 \cdot 2^{n-1} + \text{bit}_{n-2} \cdot 2^{n-2} + \cdots + \text{bit}_1 \cdot 2 + \text{bit}_0 \\ &= \text{bit}_{n-2} \cdot 2^{n-2} + \cdots + \text{bit}_1 \cdot 2 + \text{bit}_0, \end{aligned}$$

- $B = -1, \ldots, -2^{n-1}$, tj. $B < 0 \iff \text{bit}_{n-1} = 1$, i tada je

$$\begin{aligned} B &= (1 \cdot 2^{n-1} + \text{bit}_{n-2} \cdot 2^{n-2} + \cdots + \text{bit}_1 \cdot 2 + \text{bit}_0) - 2^n \\ &= (\text{bit}_{n-2} \cdot 2^{n-2} + \cdots + \text{bit}_1 \cdot 2 + \text{bit}_0) - 2^{n-1}. \end{aligned}$$

Za negativne B, bitovi bit_i nisu binarne znamenke od B.

Komplement bita

Sad ćemo izvesti jedno jednostavno pravilo o prikazu koje vrijedi (u istom obliku) za sve prikazive brojeve.

Prvo moramo uvesti pojmove komplementa jednog bita i komplementa broja (bez predznaka).

Definicija. Komplement bita (binarne znamenke) $b \in \{0, 1\}$, u oznaci $\overline{b}$, definiramo ovako:

$$\overline{b} := 1 - b.$$

Odmah vidimo da je

$$\overline{0} = 1, \quad \overline{1} = 0,$$

pa je operacija komplement aritmetički ekvivalent negacije. ■

Komplement broja bez predznaka

Definicija. Komplement prikazivog cijelog broja $B' \in \mathbb{Z}_{2^n}$ (bez predznaka) na n bitova definiramo ovako:

- nađemo prikaz broja B', tj. svih n bitova u prikazu,

$$[\, b_{n-1}\ b_{n-2}\ \ldots\ b_1\ b_0\,],$$

- komplementiramo svaki bit u prikazu, $b'_i \mapsto \bar{b}'_i$,

$$[\, \bar{b}_{n-1}\ \bar{b}_{n-2}\ \ldots\ \bar{b}_1\ \bar{b}_0\,],$$

- očitamo broj $\bar{B}'$ čiji je to prikaz. ■

Ovo možemo definirati i za brojeve s predznakom, ali nam neće trebati (da ne stvara zabunu).

Komplement broja bez predznaka (nastavak)

Dakle, ako je $B' \in \mathbb{Z}_{2^n}$ oblika

$$B' = b'_{n-1} \cdot 2^{n-1} + b'_{n-2} \cdot 2^{n-2} + \cdots + b'_1 \cdot 2 + b'_0,$$

onda je, očito, $\bar{B}' \in \mathbb{Z}_{2^n}$ i vrijedi

$$\bar{B}' = \bar{b}'_{n-1} \cdot 2^{n-1} + \bar{b}'_{n-2} \cdot 2^{n-2} + \cdots + \bar{b}'_1 \cdot 2 + \bar{b}'_0.$$

Uočimo da za svaki bit vrijedi $b + \bar{b} = 1$ i zbrojimo ove dvije relacije. Dobivamo

$$\begin{aligned} B' + \bar{B}' &= 1 \cdot 2^{n-1} + 1 \cdot 2^{n-2} + \cdots + 1 \cdot 2 + 1 \cdot 1 \\ &= 2^{n-1} + 2^{n-2} + \cdots + 2 + 1 \\ &= 2^n - 1. \end{aligned}$$

Prikaz suprotnog broja preko komplementa

Prebacimo 1 s desne na lijevu stranu

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

i zapišimo ovu relaciju "modularnim zbrajanjem" $\oplus_{2^n}$ u $\mathbb{Z}_{2^n}$

$$B' \oplus_{2^n} (\bar{B}' \oplus_{2^n} 1) = 0.$$

Dakle, jedinstveni suprotni element (= inverz obzirom na modularno zbrajanje) elementa $B' \in \mathbb{Z}_{2^n}$ jednak je $\bar{B}' \oplus_{2^n} 1$.

Drugim riječima, suprotni broj dobivamo tako da

- komplementiramo broj (odnosno, njegov prikaz) i
- dodamo 1 modulo 2^n.

Prikaz suprotnog broja preko komplementa

Napomena. Modularno dodavanje jedinice služi samo za $B' = 0$. Tada je $\bar{B}' = 2^n - 1$ i moramo dodati 1 modularno, da opet dobijemo 0 kao suprotni element.

Za $B' \in \mathbb{Z}_{2^n}$ i $B' > 0$, suprotni element je $\bar{B}' + 1$. ■

Potpuno isto "pravilo" vrijedi i za brojeve s predznakom!

Prisjetimo se da je prikaz negativnih brojeva iz $\mathbb{Z}_{2^n}^-$ "podešen" tako da modularna aritmetika radi korektno.

To znači da za svaki $B \in \mathbb{Z}_{2^n}^-$ vrijedi

$$\text{prikaz}(B) \oplus_{2^n} \text{prikaz}(-B) = 0.$$

Ovdje $-B$ treba smatrati suprotnim elementom od B u $\mathbb{Z}_{2^n}^-$.

Prikaz suprotnog broja preko komplementa

Znamo da je $\text{prikaz}(B) = B'$, za neki $B' \in \mathbb{Z}_{2^n}$, pa mora biti

$$\text{prikaz}(-B) = \bar{B}' \oplus_{2^n} 1,$$

zbog jedinstvenosti suprotnog elementa u $\mathbb{Z}_{2^n}$.

Opet, prikaz suprotnog broja $\text{prikaz}(-B)$ dobivamo tako da

- komplementiramo prikaz samog broja $\text{prikaz}(B) = B'$ i
- dodamo 1 modulo 2^n.

Ovaj zaključak je direktna posljedica modularne aritmetike u $\mathbb{Z}_{2^n}^-$ i ne ovisi o tome koji broj $B' \in \mathbb{Z}_{2^n}$ izaberemo za prikaz zadanog broja $B \in \mathbb{Z}_{2^n}^-$. Zato i treba dodatno fiksirati prikaz barem jednog broja iz $\mathbb{Z}_{2^n}^-$ (na primjer, $B = 0$ na isti prikaz kao $0 \in \mathbb{Z}_{2^n}$) da dobijemo jedinstvenost prikaza. ■

Dijeljenje cijelih brojeva s predznakom

Dijeljenje cijelih brojeva s predznakom — uvod

Ranije smo uveli operacije div (cjelobrojni kvocijent) i mod (ostatak) na skupu $\mathbb{Z} \times \mathbb{N}$.

Definicija. Neka su $a \in \mathbb{Z}$ i $b \in \mathbb{N}$ bilo koji brojevi, i neka su $q \in \mathbb{Z}$ (cjelobrojni kvocijent) i $r \in \mathbb{Z}_b$ (ostatak) jedinstveni brojevi za koje vrijedi

$$a = q \cdot b + r.$$

Operacije div i mod definiramo relacijama

$$a \text{ div } b := q \in \mathbb{Z}, \quad a \text{ mod } b := r \in \mathbb{Z}_b.$$

Za početak, uočite da su obje operacije definirane na skupu $\mathbb{Z} \times \mathbb{N}$, a kodomene su im različite.

Dijeljenje cijelih brojeva s predznakom — uvod

Sad nam treba proširenje na skup $\mathbb{Z} \times (\mathbb{Z} \setminus \{0\})$,

- kad cjelobrojno dijelimo dva cijela broja (što, naravno, ima smisla),

da dobijemo cjelobrojno dijeljenje za cijele brojeve s predznakom $\mathbb{Z}_{2^n}^{-}$ (koji modeliraju skup $\mathbb{Z}$).

Naravno, ideja je ista kao i kod brojeva bez predznaka.

Cjelobrojno dijeljenje ili dijeljenje s ostatkom cijelih brojeva s predznakom je naprosto

- restrikcija odgovarajućih operacija div i mod.

Tek u novije vrijeme postoji dogovoreni standard (tzv. C99)

- za proširenje operacija div i mod na $\mathbb{Z} \times (\mathbb{Z} \setminus \{0\})$.

Dijeljenje cijelih brojeva — što je problem?

Primjer. Neka je $a = 5$ i $b = 3$. Onda je

- $5 = 1 \cdot 3 + 2$, pa je kvocijent $q = 1$ i ostatak $r = 2$.

Neka je sad $a = -5$ i $b = 3$. Za kvocijent q i ostatak r mora vrijediti $a = q \cdot b + r$. Ostaje izbor/restrikcija ostatka r.

U Euklidovom teoremu je uvijek $r \geq 0$, tj. $r \in \mathbb{Z}_b$. Onda je

- $-5 = -2 \cdot 3 + 1$, pa je $q = -2$ i $r = 1$.

To nema "nikakve veze" s prethodnim rezultatima za pripadne apsolutne vrijednosti!

Međutim, ako dozvolimo negativni ostatak $r \in -\mathbb{Z}_b$, onda je

- $-5 = -1 \cdot 3 - 2$, pa je $q = -1$ i $r = -2$.
- Apsolutne vrijednosti kvocijenta i ostatka ostaju iste! ■