Programiranje 1
4. predavanje

Saša Singer

singer@math.hr

web.math.hr/~singer

PMF – Matematički odjel, Zagreb

Sadržaj predavanja

- Osnovni tipovi podataka u računalu (pregled):
  - Višekratnici byte-a, odn. riječi (bez strukture).
  - Cijeli brojevi bez predznaka.
  - Cijeli brojevi s predznakom.
  - “Realni” (floating–point) brojevi.
- Cijeli brojevi — prikaz i aritmetika:
  - Prikaz brojeva bez predznaka — sustav ostataka.
  - Modularna aritmetika cijelih brojeva.
  - Adrese — “obična” aritmetika.
  - Prikaz brojeva s predznakom — sustav ostataka.
  - Tipične pogreške u korištenju cijelih brojeva.

Odrade za 2. 11.

U petak, 2. studenog — nema nastave iz Programiranja 1.

To ćemo odraditi u subotu, 10. studenog, po "normalnom" rasporedu za petak:

- predavanja (S. Singer) — 10–12 u (003),
- vježbe (V. Šego) — 12–14 u (004).

Ostatak "moje" grupe ima vježbe (M. Doko) u srijedu 31. listopada, prema rasporedu (bez odrade).

Pripremite se za dolazak "na brdo"!

Informacije

Ne zaboravite da treba:

- otvoriti korisnički račun u Računskom centru,
- utorkom i četvrtkom, od 11 do 13 sati.

Nadalje, treba:

- obaviti prijavu i dobiti potvrdu prijave

u aplikaciji za tzv. “domaće zadaće”, na web–adresi

`http://degiorgi.math.hr/prog1/ku/`

Prilikom prijave za “ku”, svoje podatke trebate upisati korektno, što (između ostalog) znači i

- korištenje hrvatskih slova u imenu i prezimenu!

Informacije — nastavak

Studenti koji su upisali "czsdj" varijantu imena i prezimena neka se jave e-mailom asistentu V. Šegi na adresu

`vsego@math.hr`

i napišu

- svoj JMBAG i ispravno ime i prezime.

Tom prilikom, zaista nije nužno da (onako usput)

- pošaljete i svoju lozinku, tj. password!

Upravo suprotno: nemojte to raditi!

Napomene o sigurnosti — lozinka (password)

Za početak, za sve što se radi pod nekim korisničkim računom ili accountom,

- odgovoran je vlasnik tog računa!

Ne vrijedi isprika da vam se netko drugi logirao umjesto vas!

A sve što vam treba za logiranje na račun su dvije stvari:

- korisničko ime ili username — koji se vidi kad ga kucate,
- lozinka iliti password — koji se ne vidi, i to s razlogom.

Zapamtite: password vam je

- jedina zaštita od "nezvanih" korisnika.

Napomene o sigurnosti (nastavak)

Zato nemojte koristiti "početni" password, već ga

- promijenite na nešto "tajno", što zaista samo vi znate.

I, na kraju rada,

- uvijek treba uredno završiti rad, tj. "odlogirati" se.

Što mislite — kako funkcionira "on–line" kupovina!?

Prikaz podataka u računalu

Sadržaj

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

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 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.

Napomena: kasnije smo pojam "riječ" koristili za nešto drugo — prostor za cijele brojeve, odnosno, instrukcije.

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,
- 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

$$1 \text{ byte} = 8 \text{ bitova}.$$

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

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

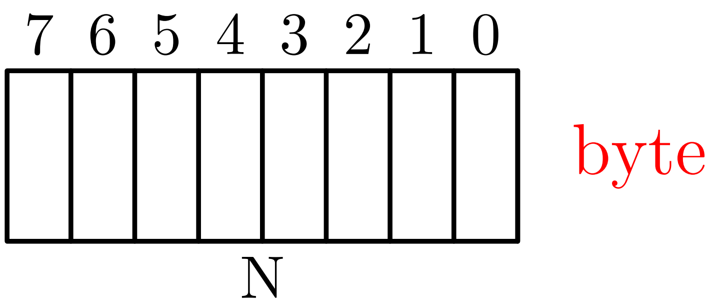
Naziv	Duljina (bitova)	Broj byteova
word	16	2
doubleword	32	4
quadword	64	8
double quadword	128	16

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.

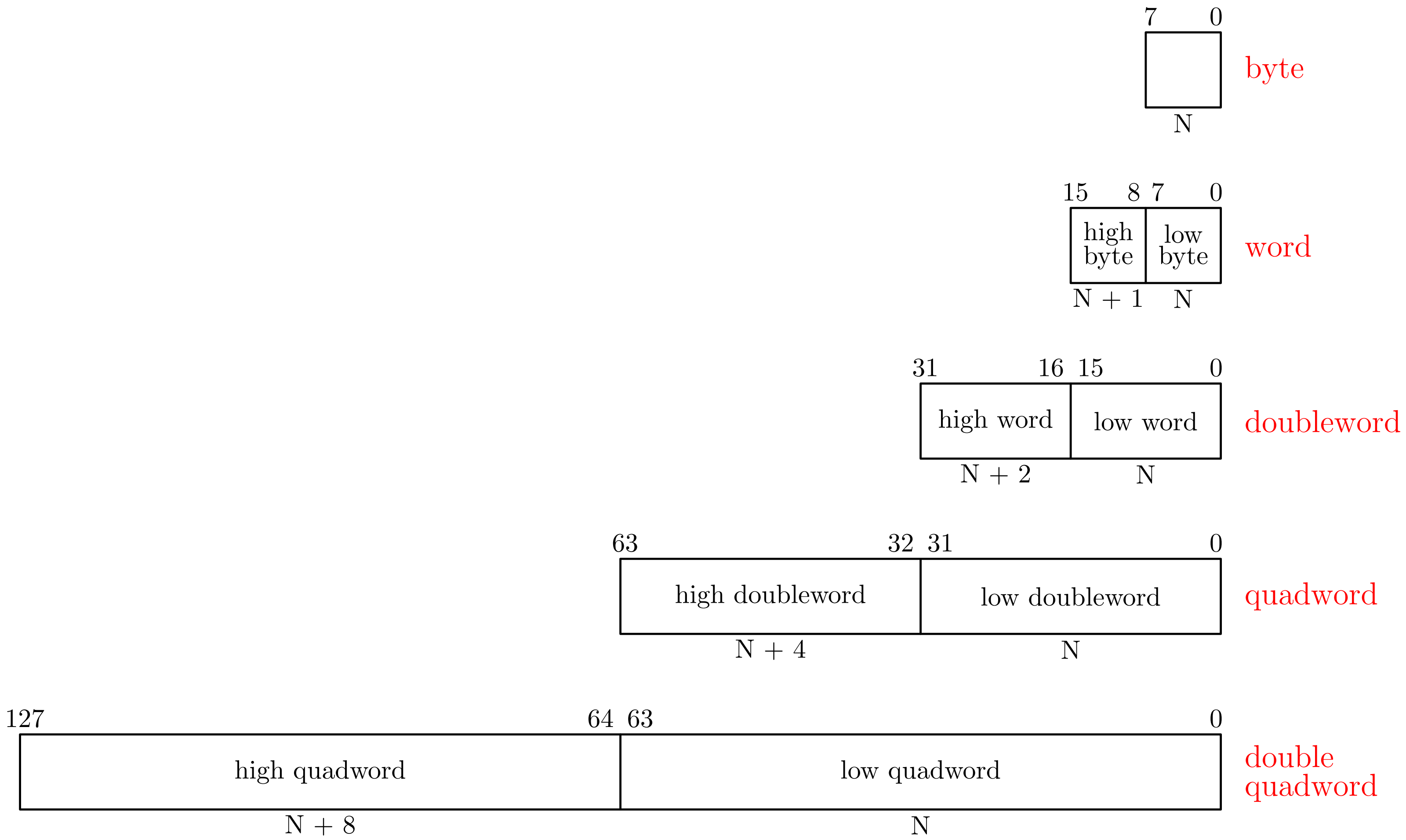
Detalji malo kasnije!

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 byteovi (adresabilne cjeline na IA–32 su byteovi).
- Na IA–32, najniži byteovi su i na najnižim adresama, ali postoje arhitekture računala na kojima je obratno (vodeći dio je na najnižoj adresi).
- Dogovorno, najniža adresa (na kojoj "počinje" cjelina) je ujedno i adresa čitave cjeline (bez obzira na poredak dijelova).

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.

Logičke vrijednosti (primjer)

C nema posebni tip za logičke vrijednosti.

```
#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.

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:

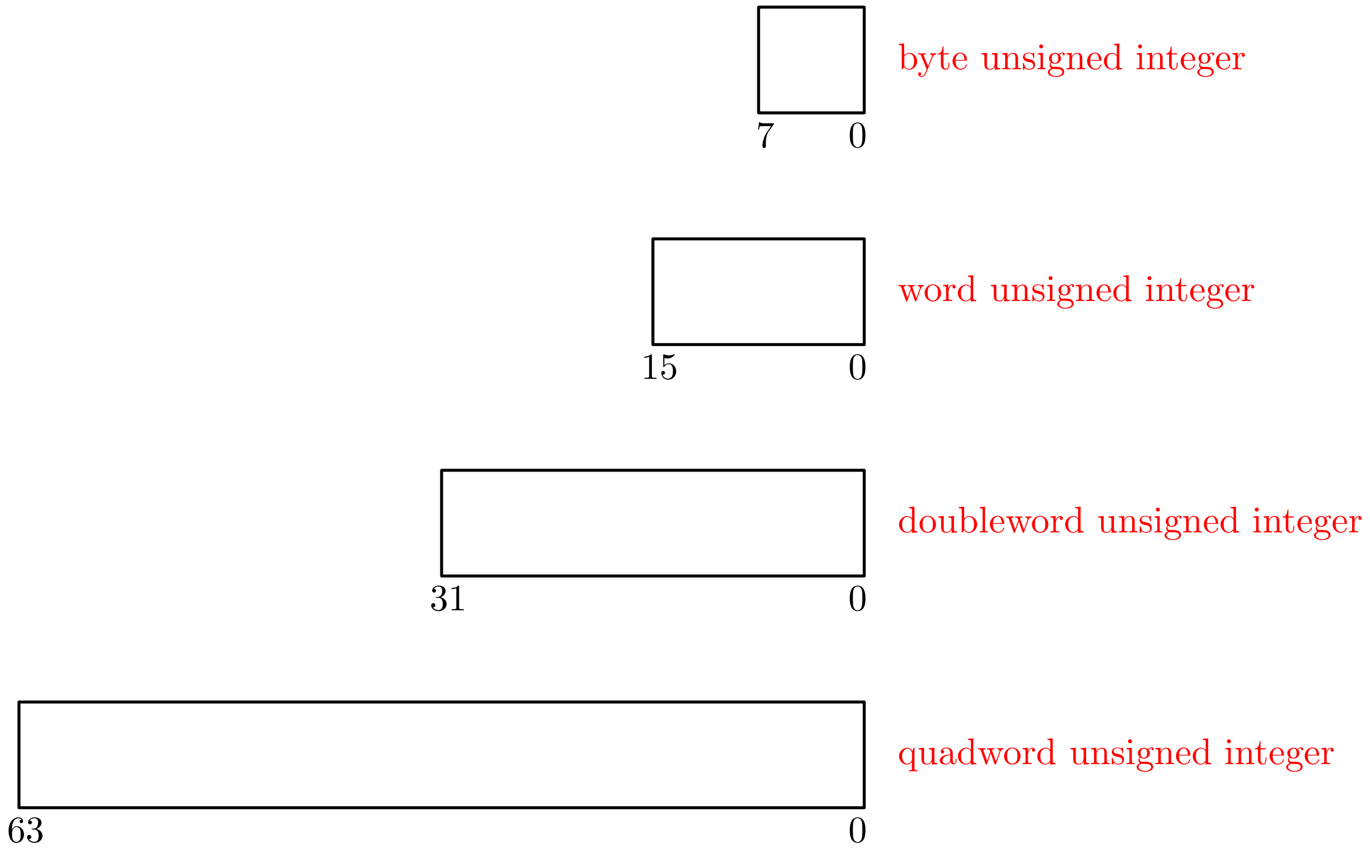
- koji konačni skupovi brojeva modeliraju odgovarajuće matematičke skupove,
- kako se točno prikazuju njihovi elementi u računalu,
- kako se realizira artimetika na tim skupovima.

Detalji u nastavku.

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

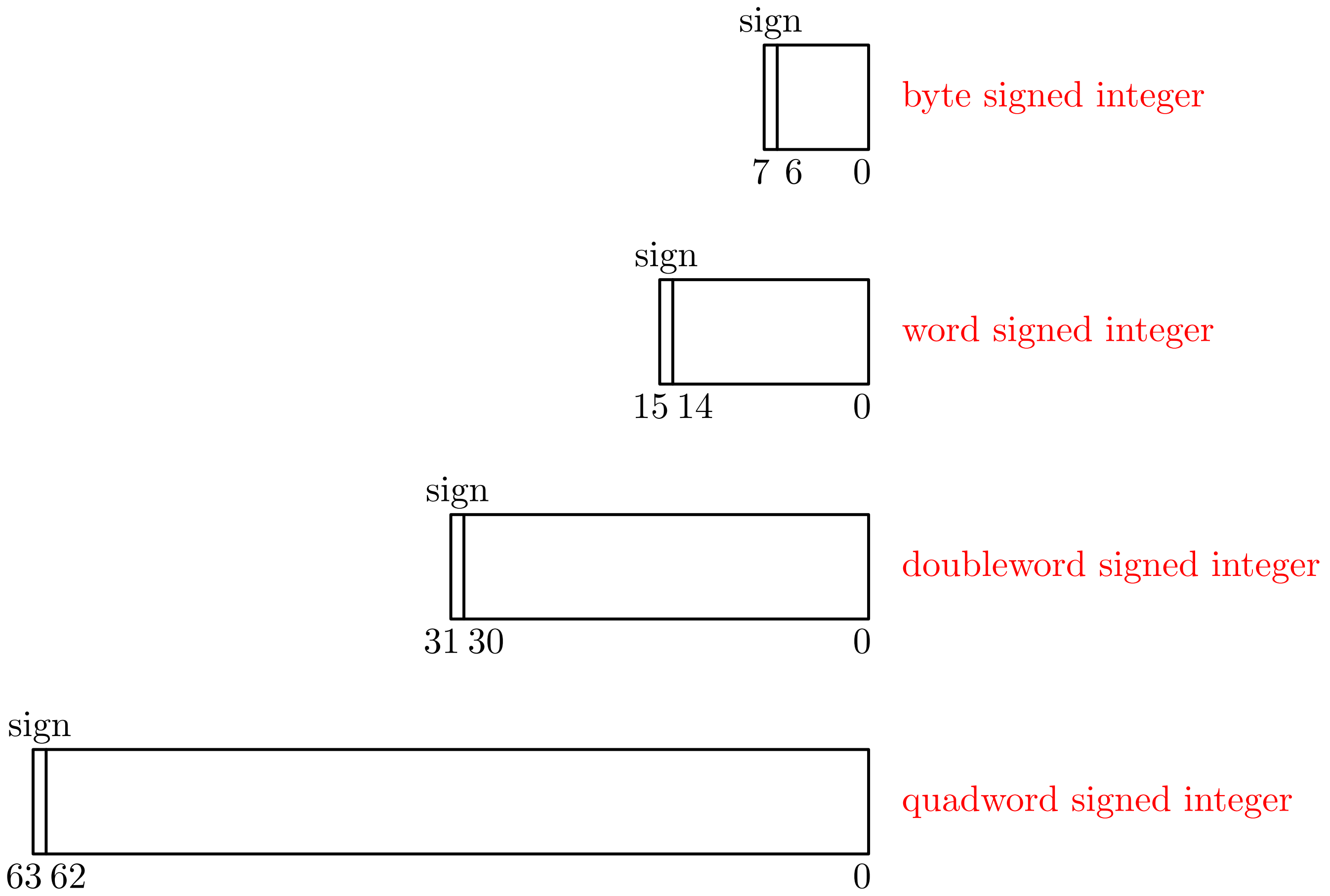
Numerički tipovi podataka na IA–32

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



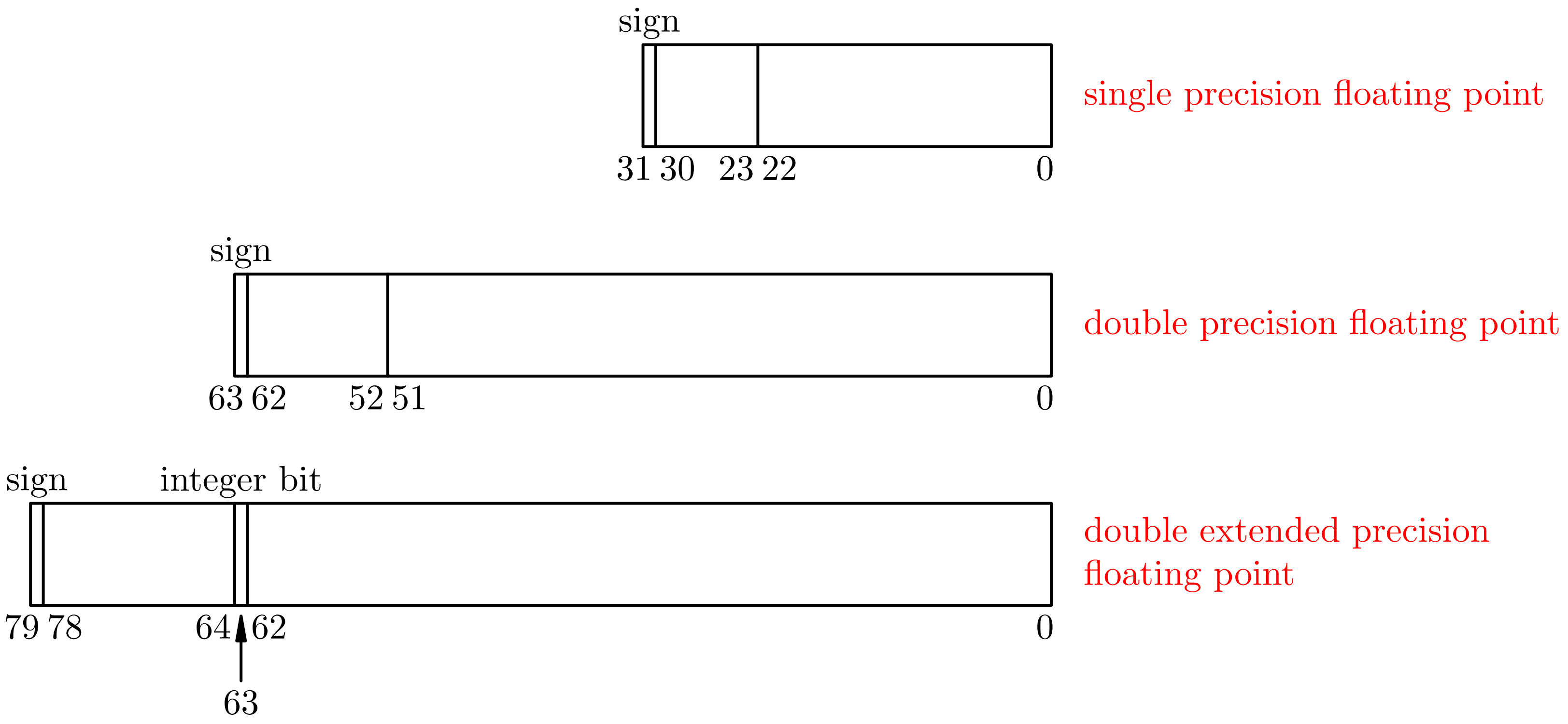
Numerički tipovi podataka na IA–32 (nastavak)

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 predavanja

- Cijeli brojevi — prikaz i aritmetika:
  - Prikaz brojeva bez predznaka.
  - Modularna aritmetika cijelih brojeva.
  - Aritmetika adresa — "obična" aritmetika.
  - Prsten ostataka modulo 2^n.
  - Dijeljenje s ostatkom — Euklidov teorem.
  - Prikaz brojeva bez predznaka — sustav ostataka.
  - Prikaz brojeva s predznakom — sustav ostataka.
  - Prikaz negativnih brojeva — komplementiraj i dodaj 1.
  - 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 (nezavisno od ostalih) biti jednak 0 ili 1.

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 prikazivih brojeva ima 2^n elemenata, pa je on jednak

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

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

Kako stvarno izgleda prikaz brojeva u tih n bitova?

Prikaz pojedinih (prikazivih) brojeva je

- doslovna "kopija" prikaza tog broja u pozicionom zapisu u bazi 2.

Što to znači?

Prikaz cijelih brojeva bez predznaka

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 nemamo "previše" nul–znamenki "sprijeda".

Prikaz cijelih brojeva bez predznaka (nastavak)

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

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

Prikaz cijelih brojeva bez predznaka (nastavak)

U računalu će to biti prikazano kao “cjelina” od n bitova

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

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

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

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

Prikaz cijelih brojeva bez predznaka — primjer

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

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

pa je 123 prikaziv. Nadalje, njegov binarni prikaz je

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

Dakle, broj 123 kao cijeli broj bez predznaka ima prikaz

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

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 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.

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 "pravim" korisničkim podacima, 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 (aritmetika pokazivača ili pointera u C-u), posebno kod obrade nizova.
- 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).

Aritmetika adresa (nastavak)

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

Primjer. Ne znam da li znate da MS Windows XP ima "ograničenje" adresa na samo 2 GB, što je polovina od normalnog adresnog prostora na IA–32 (4 GB).

Stvar ide tako daleko da se XP "ne diže" ako u računalu imate više od 2 GB memorije (probao sam).

U čemu je "štos" — ne znam. Intel kaže da je ograničenje od 2 GB "tvrdo" ugrađeno u osnovne Microsoftove razvojne biblitoeke (tzv. SDK), koje svi proizvođači (pa i Intel) koriste za razvoj svojih programa (recimo, C i Fortran compilera).

Ako netko sazna kako nagovoriti XP da uredno "vidi" više od 2 GB memorije, javite mi.

Aritmetika cijelih brojeva bez predznaka (opet)

Zašto se aritmetika realizira na ovaj način, kao modularna aritmetika?

Postoje dva vrlo dobra razloga.

- Čisto tehnički, ova realizacija je brza.

Ostaci modulo 2^n (uz ignoriranje prijenosa) znače da stalno uzimamo samo najnižih n bitova rezultata, što je lako realizirati.

Drugi razlog je matematičke prirode.

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

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

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 $\mathbb{Z}_{2^n}$, koje se upravo na taj način realiziraju u računalu za cijele brojeve bez predznaka.

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

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 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

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

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 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 $\mathbb{N}_0$).

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).

Prsten ostataka modulo b

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 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 brojevim.

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

- restrikcija "cjelobrojnih" operacija div i mod.

Dijeljenje cijelih brojeva bez predznaka

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

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

Dijeljenje cijelih brojeva bez predznaka

Naravno, cijela stvar vrijedi i za $b = 2^n$.

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.

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 ostataka modulo b

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 svaka dva broja A, $B \in \mathbb{Z}_b$, $B > 0$, 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 ostataka modulo b

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

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

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

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

Dakle, “modularni” kvocijenti su 1, 3, 5 i 7, a najmanji je pravi.

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.

Razlike su u broju bitova n predviđenih za prikaz.

- Zapis konstanti (vrijednost, navođenje tipa).
- 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;
}
```

Cijeli brojevi u C-u — čitanje

Primjer C programa za prikaz brojeva i čitanje brojeva.

Kod čitanja, pretvorba iz niza znakova (dekadske znamenke u dekadskom zapisu) u binarni zapis ide onim aritmetičkim pravilima koja odgovaraju tipu broja ("Hornerov" algoritam).