Programiranje 1
2. predavanje

Saša Singer

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

PMF – Matematički odjel, Zagreb

Sadržaj predavanja

- Uvod u algoritme:
  - Pojam algoritma.
  - Primjeri algoritama.
  - Opis algoritma.
  - Zapis i izvršavanje algoritama.
  - Osnovna svojstva algoritma.
- Principi rada računala:
  - Što je računalo?
  - Instrukcije — operacije i podaci.
  - Osnovni dijelovi računala — ulaz, izlaz, memorija.
  - Izvršni dio računala — procesor.
  - Von Neumannov model računala.

Sadržaj predavanja (nastavak)

- Građa računala:
  - Memorija (bistabil, bit, riječ, byte),
  - Procesor (registri, aritmetičko–logička jedinica, upravljačka jedinica),
  - Ulazna jedinica, izlazna jedinica.

Informacije

Prijedlog odrade predzadnjeg predavanja (17. 12.):

- u subotu, 25. 9., od 10–12 u (003).

Informacije

Ne zaboravite da treba:

- otvoriti korisnički račun u Računskom centru.
- Računi se "preuzimaju" u centru, utorkom i četvrtkom, od 12:30 do 15 sati.

Promijenite password!

Nadalje, treba:

- obaviti prijavu i dobiti potvrdu prijave

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

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

Informacije — nastavak

Bitno: Prilikom prijave za "ku",

- svoje podatke trebate upisati korektno,

što (između ostalog) znači i

- korištenje hrvatskih znakova u imenu i prezimenu!

Uvod u algoritme

Sadržaj

- Uvod u algoritme:
  - Pojam algoritma.
  - Primjeri algoritama.
  - Opis algoritma.
  - Zapis i izvršavanje algoritama.
  - Osnovna svojstva algoritma.

Pojam algoritma

Što je algoritam? Grubo rečeno:

- Algoritam = metoda, postupak, pravilo za rješenje nekog problema ili dostizanje nekog cilja.

Ovo nije precizna definicija u matematičkom smislu, već samo opis preko drugih, sličnih pojmova, pri čemu je "postupak" najbliži.

- Postupak asocira na konačan niz koraka koje treba napraviti za rješenje nekog problema.
- Metoda se kao izraz često koristi u matematici, ali, obično, uključuje i tzv. beskonačne "postupke" — koji tek na limesu daju rješenje (v. matematička analiza, numerička matematika).

Pojam algoritma (nastavak)

Zašto je bitno znati što je algoritam?!

- Osnovna zadaća: razvoj efikasnih i točnih algoritama.
- Intuitivno je jasno da efikasno znači brzo, a točno da je rješenje blizu "pravom" rješenju. Detaljnije — poslije.

Postoji i precizna matematička definicija pojma algoritam, ali ona nije sasvim jednostavna — potrebna je onima koji će se baviti svojstvima algoritama.

Budući da je cilj kolegija

- naučiti koristiti algoritme za rješavanje raznih problema,

dovoljno je (umjesto definicije)

- opisati osnovna svojstva algoritama.

Primjer algoritma 1

Sva moderna tehnička pomagala imaju upute za uporabu, korištenje, rukovanje, instalaciju, ... ili kako vam drago.

Primjer. Dojadilo mi je slušanje zaštićenih glazbenih CD-ova na računalu, jer:

- prvo se javi neki odurni "player",
- a nakon toga počne glazba kôdirana u MP3 formatu nevjerojatne nekvalitete od 56 KB u sekundi.

Prvo rješenje: tehničko i nije za javno predavanje.

Drugo rješenje: kupio sam priglupu glazbenu CD liniju! Inače se može dogoditi sličan problem kao i na računalu — da stvar ne radi. Recimo, takvi su CD-playeri za automobile.

Primjer algoritma 1 (nastavak)

I gdje je tu algoritam?

Što dolazi uz glazbenu liniju?

- Upute = kako pospajati sve razne dijelove, kablove i ostalo u funkcionalno radeću stvar!

Dakle, dotične upute su:

- algoritam za postizanje cilja = kako iz hrpe dijelova sastaviti radeću glazbenu liniju.

One "druge" upute za uporabu imaju smisla tek kad završimo s ovima za "instalaciju"!

Primjer algoritma 1 (nastavak)

To može izgledati ovako:

- Kabel A (slijedi sličica) treba izvaditi iz vrećice,
- utaknuti otraga u CD jedinicu u rupu B (vidi sličicu),
- pažljivo stisnuti konektore,
- zavinuti kabel,
- prisloniti ga na stražnju površinu CD-jedinice
- i na kraju ga (ipak) utaknuti u rupu C na pojačalu (nova sličica).
- ...

Sad se sličan postupak ponavlja jedno desetak puta!
Za sat–dva, možda linija i proradi!

Još primjera algoritama

Standardni primjeri algoritama:

- kulinarski recepti (pripravljanje jela),
  - pita od jabuka, muffini,
- recepti za pripravu pića i koktela,
  - "bijeli medvjed" (oprez ...),
- uporaba bilo kakvih aparata (izbor operacija, ...),
- rješavanje matematičkih zadataka!

Opis algoritma

Općenito, kako moraju izgledati upute i iz čega se sastoje?

- Opći izgled je nešto poput:

  1. korak
  2. korak
  ⋮
  zadnji korak

Drugim riječima, algoritam se sastoji iz niza koraka koje treba izvršiti da bi se postigao cilj, odnosno, riješio problem.

Tih koraka, naravno, ima konačno mnogo.

- Svaki pojedini korak algoritma je instrukcija ili naredba (može i akcija) koju treba napraviti (izvršiti).

Zapis i izvršavanje algoritama

Uobičajeno, instrukcije se

- pišu jedna ispod druge i
- izvršavaju tim redom.

Ova "očita" pravila vrijede i u programiranju!

Ipak, postoje i instrukcije koje

- mijenjaju standardni redoslijed izvršavanja.

Primjeri:

- ako se dogodi "to i to", onda otiđi na korak "taj i taj", ili
- ponovi "neke korake" sve dok je ispunjen uvjet "taj i taj", ili
- ponovi "neke korake" određeni broj puta.

Vrste instrukcija

Grubo govoreći, postoje dvije vrste instrukcija za kontrolu redoslijeda operacija, tj. izvršavanja ostalih intrukcija:

- Uvjetne = izbor jedne od mogućih alternativa.
- Petlje = ponavljanje nekog bloka naredbi pod kontrolom uvjeta ili brojača.

Ovakve instrukcije su ključne u programiranju.

Razlog?

Bez petlji, programiranje nema smisla. Poanta:

- Program je "kratak", a izvršavanje "traje", tj. dijelovi programa se ponavljaju puno puta.
  (Računalo je "glupo, ali brzo".)

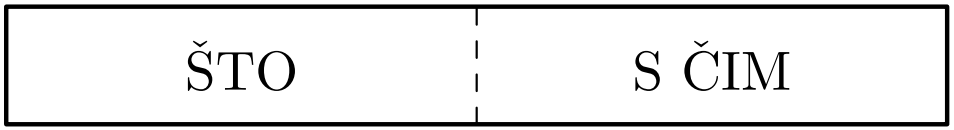
Izgled instrukcije

Sve instrukcije imaju sličan oblik i sastoje se iz 2 dijela:

- što treba napraviti = operacija,
- nad čim to treba izvršiti = objekt nad kojim se obavlja operacija.

Primjer. Izvadi kabel (sličica) iz vrećice.

Isti oblik imaju i instrukcije na računalu:



Primijetite da se ista operacija

- može obavljati na(d) raznim podacima.

Opis algoritma — općenitost

Algoritam bi trebao raditi nad "općenitim" podacima, tj. bitno je da se samo radi o istom postupku.

- Konkretne podatke zadajemo svaki put kad izvršavamo algoritam, tj. možemo ih mijenjati.

Na primjer, puno je bolje napisati algoritam koji nalazi rješenja "opće" kvadratne jednadžbe

$$ax^2 + bx + c = 0,$$

za proizvoljne a, b, $c \in \mathbb{R}$, uz $a \neq 0$, nego onaj koji nalazi rješenja "konkretne" jednadžbe

$$x^2 - 3x + 2 = 0.$$

(Rješenja su, očito, 1 i 2.)

Opis algoritma (nastavak)

U slučaju kvadratne jednadžbe, rješenja su dana formulom:

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

Primijetite da izračunata rješenja mogu biti kompleksna.

- Što su ovdje instrukcije?

Napomene o točnosti (bolje rješenje — kasnije)!

Rješivost jednadžbi višeg stupnja:

- Do stupnja 4 ide "formulama",
- preko 5 — nema općih formula, tj. moramo koristiti približne numeričke metode.

Svojstva algoritma

Dakle, algoritam izvodi neke operacije nad nekim podacima, u obliku niza koraka, i daje neko rješenje.

Gruba skica:

Osnovna svojstva algoritma su:

- ima (ili nema) ulazne podatke,
- ima izlazne podatke,
- završava u konačnom vremenu,
- uvijek je nedvosmisleno definiran,
- mora biti efikasan, tj. završiti u razumnom vremenu.

Ulaz/izlaz

Ulaz:

- Svaki algoritam ima 0 ili više, ali konačno mnogo ulaznih podataka (inače ih ne možemo ni pročitati).

Ako ih ima, njih moramo zadati/izabrati iz neke dozvoljene klase ulaznih objekata.

Algoritmi s 0 ulaza nisu jako česti, ali ima ih (obično, baš u matematici). Opisuju fiksni postupak.

- Recimo: provjeri je li 327 prost broj (i, je li?), ili riješi konkretnu kvadratnu jednadžbu.

Ako algoritam ima više dozvoljenih objekata na ulazu, kažemo da je općenit, jer rješava cijelu klasu problema.

- Recimo: kvadratna jednadžba s općim a, b i c.

Ulaz/izlaz (nastavak)

Izlaz:

- Svaki algoritam mora imati bar jedan izlaz, jer inače nije ostavio trag svog izvršavanja.

To je traženo "rješenje" našeg problema.

Napomena. Postoje programi koji se "stalno vrte", bez izlaza. Njih zovemo procesi. Na primjer, operacijski sustav računala.

Ova prva dva svojstva pričaju o objektima na ulazu i izlazu, ali ništa ne kažu o tome kako se iz jednog dolazi do drugog.

Ostala svojstva nešto govore o "crnoj kutiji" na ranijoj skici

Konačnost

Konačnost:

- Svaki algoritam mora završiti u konačno mnogo koraka, za svaki dozvoljeni ulaz.

U programu uvijek treba provjeriti je li ulaz korektno zadan.

- Na primjer, u kvadratnoj jednadžbi, koeficijent a može biti 0. U tom slučaju, jednadžba nije kvadratna, a u formuli za rješenje pojavit će se dijeljenje s 0!

Savjet. U praksi treba predvidjeti sva moguća korisna ograničenja ulaza i ugraditi ih u program.

- Na primjer, program koji učitava temperaturu T vode u nekoj posudi, trebao bi provjeriti je li $0 \leq T \leq 100$.

Brojevi u prirodi imaju jedinice — ovdje je T u °C.

Definiranost i nedvosmislenost

Kad projektiramo algoritam, ne znamo odmah na početku

- od kojih se koraka sastoji čitav postupak za rješenje problema.

Obično se problem rastavi na nekoliko većih cjelina, koje rješavamo pazeći na međuovisnost potproblema.

- Ako su te cjeline prevelike za izravno rješavanje, one se mogu rastavljati na još manje dijelove, ...

Ova metoda zove se metoda postupnog profinjavanja (engl. stepwise refinement).

Dokle treba postupno profinjavati?

- Ovisi o izvršitelju algoritma — koje instrukcije on "prepoznaje", u smislu da ih može ili zna izvršavati.

Definiranost i nedvosmislenost (nastavak)

Definiranost i nedvosmislenost:

- Algoritam se sastoji od niza osnovnih (elementarnih, primitivnih) instrukcija, koje moraju biti jednoznačno i nedvosmisleno definirane za izvršitelja algoritma.

Primjer. Ja volim tropetinsku pitu od jabuka (3/5 jabuka i 2/5 tijesta).

Ako je moja "bolja polovica" kod kuće, onda će moja molba "Napravi mi pitu od jabuka" biti uslišena.

- Za nju je pravljenje pite od jabuka elementarna instrukcija, jer je ona dovoljno moćan izvršitelj.

Dakle, za moj algoritam "kako se dočepati pite" to je elementarna instrukcija.

Definiranost i nedvosmislenost (nastavak)

Naravno, postoji i lošije rješenje (po mene).

Ako ona nije u blizini, a ja ipak želim pitu od jabuka, onda je

- instrukcija "ispeci pitu od jabuka" za mene presložena,
- pa moram zaviriti u kuharicu, gdje su dane "jednostavne" instrukcije za pravljenje pite (vidi prilog).

Kad za rješavanje problema koristimo računalo, potrebno je znati

- što ono "zna" i može napraviti,
- tj. što su elementarne instrukcije — operacije i podaci.

Za zapisivanje tih algoritama postoje jezici ključno određeni "snagom" računala (poput C-a).

Efikasnost

Efikasnost:

- Algoritam mora završiti u razumnom vremenu, što je bitno jači zahjev od konačnosti.
- Recimo, 500 godina nije razumno vrijeme!

A, ima li takvih algoritama? Ima!

- Tzv. NP–potpuni problemi, za koje se ne zna postoje li efikasni algoritmi. (Otvoreni problem "P = NP?".)
- Primjer. Problem trgovačkog putnika (TSP).

Napomena. Postoje problemi za koje ne postoji algoritam za njihovo rješenje — tzv. algoritamski nerješivi problemi.

Primjer. Popločavanje ravnine (skiciraj).

Principi rada računala
(Građa i funkcioniranje)

Sadržaj

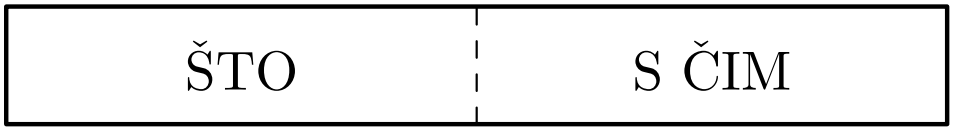
- Principi rada računala:
  - Što je računalo?
  - Instrukcije — operacije i podaci.
  - Osnovni dijelovi računala — ulaz, izlaz, memorija.
  - Izvršni dio računala — procesor.
  - Von Neumannov model računala.

Što je računalo?

Što je računalo?

- Računalo = stroj za izvršavanje algoritama.

Prisjetimo se općeg oblika instrukcija:



što = operacija, s čim = podatak ili operand.

Operacije koje računalo "zna" izvršiti su

- osnovne strojne instrukcije.

Podaci s kojima "zna" raditi su

- osnovni ili elementarni podaci.

Građa i funkcioniranje računala

- Želja: imati računalo koje dozvoljava složene instrukcije na složenim podacima.
- Problem: čisto tehnološki kako to brzo realizirati. Zbog toga postoje ograničenja koja diktiraju opći izgled današnjih računala.
- Zašto matematičar uopće mora nešto znati o principima rada i građi računala?

  Zato da bi ih mogao efikasno koristiti, pisati brze i točne algoritme (ograničenja proizlaze iz fizičke građe).

Ulaz, izlaz, memorija

- Budući da algoritam ima ulaz i izlaz, mora to imati i računalo.
- Ulazni dio: čita podatke s nekog medija.
- Izlazni dio: mehanizam koji će na neki medij napisati podatke.
- Algoritam izvršava neke instrukcije nad podacima koje je učitao, tj. iz njih pravi nove podatke koje može više puta koristiti. Prema tome, računalo mora moći te podatke negdje spremiti — treba imati neku memoriju u koju može spremiti podatke i iz nje čitati.

Izvršni dio računala

- Potrebno imati i izvršni dio — koji izvršava instrukcije nad podacima.
- Dilema: treba li mi poseban stroj za svaki algoritam posebno, ili je bolje imati stroj opće namjene?
- Poseban stroj izvršava samo jedan algoritam koji je tvrdo ugrađen u arhitekturu stroja. Nije besmisleno imati takav stroj. Ako malo šire gledamo, to nas asocira na upravljačke sklopove u ulazno–izlaznim jedinicama poput CD čitača ili "pržilice".
- Računalo opće namjene je složenije, jer s algoritmom mora postupati slično kao s podacima. Svaki algoritam mora učitati, spremiti, izvršiti.

Izvršni dio računala (nastavak)

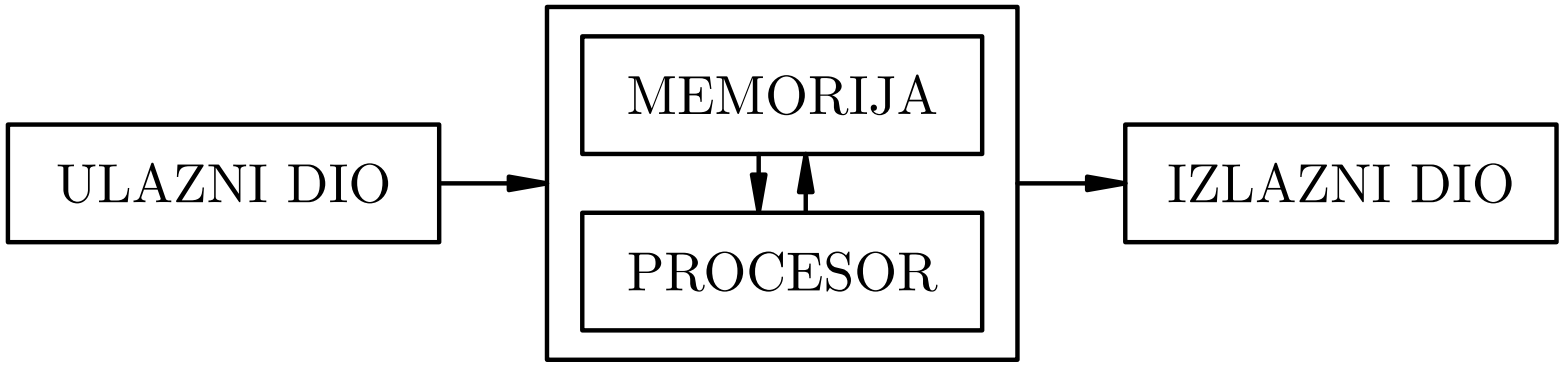
- Svako računalo opće namjene stalno izvršava jedan meta-algoritam (jer se stalno vrti – ne završava izvođenje dok ne ugasimo stroj). To je ono što obično zovemo operacijski sustav.
- Pitanje: gdje se spremaju algoritmi, odnosno programi?
- Mogli bismo imati posebnu memoriju za programe, ali nema potrebe za tim, pa se programi spremaju u istu memoriju kao i podaci.
  - Matematički "model" računala — tzv. Turingov stroj (v. dodatak), ima upravljački modul.
- Spremanje podataka i algoritama u istu memoriju ključna je stvar von Neumannovog modela računala.

von Neumannov model

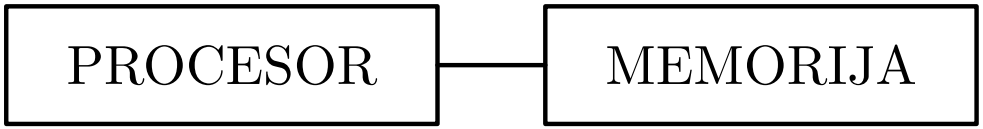
- Ideja je nastala početkom četrdesetih godina prošlog stoljeća, a objavljena je tek nakon 2. svjetskog rata.
- Za vrijeme rata poslužila je kao podloga za gradnju prvih računala opće namjene, koja su računala balističke tablice (posebno za brodove) i jasno je da je bila strogo čuvana tajna.

Izvršni dio računala (nastavak)

- Sasvim općenito, ako izvršni dio računala zovemo standardnim imenom – procesor, onda shematski model računala izgleda ovako:



- Skraćena slika bitnog dijela računala:



Građa računala
(Osnovi dijelovi računala)

Sadržaj

- Građa računala:
  - Memorija (bistabil, bit, riječ, byte),
  - Procesor (registri, aritmetičko–logička jedinica, upravljačka jedinica),
  - Ulazna jedinica, izlazna jedinica.

Memorija — bistabil i bit

- Memorija se sastoji od osnovnih elemenata koje zovemo bistabili.
  - Bistabil može biti u (jednom od) 2 stabilna stanja (BI = 2).
- Stabilno stanje? Ako je element u jednom od stanja, on će ostati u tom stanju sve dok ne uložimo energiju da se to stanje promijeni u drugo stanje.
- Matematički rečeno, količina informacije koju možemo spremiti (pohraniti) u takvom elementu je

  1 bit = 1 binarna znamenka.

  Zato se ta stanja uobičajeno i označavaju binarnim znamenkama 0 i 1.

Memorija — malo povijesti

- Nekad, u doba ranih računala (1960-ih) bistabil se realizirao pomoću feritnih jezgrica.
  - Feritne jezgrice sastojale su se od sitnih željeznih prstenova kroz koje je prolazila žica.
  - Puštanje struje u jednom ili drugom smjeru rezultiralo je magnetizacijom te jezgrice u jednom od 2 smjera.
- Danas se memorija izrađuje od sitnih tranzistora koji rade kao elektronički prekidači, po principu
  - ima struje — nema struje,

  tj. opet imaju 2 stanja.
- Sutra . . . Tko zna?

Memorija — bitovi i logičke operacije

- Osnovne logičke operacije ne, i, ili na pojedinim bitovima realiziraju se tzv. logičkim sklopovima.
- Uočite da logičke operacije "rade" kao aritmetičke na binarnim znamenkama 0, 1:
  - ne — komplement, ili $1 - \text{operand}$, ili $0 \leftrightarrow 1$,
  - i — množenje,
  - ili — zbrajanje.
- Kad bitove organiziramo u veće cjeline (na primjer, dogovor prikaza cijelih brojeva binarnim znamenkama),
  - pomoću takvih logičkih sklopova mogu se realizirati i osnovne aritmetičke operacije na brojevima.

  (Zbrajač i slični "elektronički sklopovi".)

Memorija — zašto bitovi?

Čisto tehnički, tu postoje 2 bitna ograničenja:

- Nemoguće je napraviti brzi stabilni element koji bi imao više od 2 stabilna stanja.
  - Bilo je nekih pokušaja s 3,
  - a cijela stvar je počela mehanički s 10.

  No, to je presporo. Zato su računala "binarna".
- Brzina svjetlosti je, trenutno, fundamentalno ograničenje brzine računala.
  - Minijaturizacija — 130 nm, 90 nm, 65 nm, 45 nm, ... tehnologije, uglavnom, služi povećanju brzine.

  Tu smo stigli blizu granice, s današnjom tehnologijom.

Memorija — "usko grlo" računala

Brzina svjetlosti diktira brzinu upravljanja i izvršavanja operacija u računalu (puštanje struje kroz vodiče, a onda sve ovisi o tome jesu li prekidači otvoreni ili ne).

- Logički sklopovi u procesoru su još relativno brzi. Na primjer,
  - standardni procesori rade na frekvencijama od preko 3 GHz.

  Međutim, najveća frekvencija je 3.8 GHz i tu stoji već neko vrijeme. (Tzv. Mooreov zakon više ne vrijedi.)
- Daljnji napredak u snazi procesora ne ide ubrzanjem, nego paralelizacijom (Dual core, Quad core, ...).

Memorija — "usko grlo" računala (nastavak)

- Kod memorija, situacija je kompliciranija, jer je bistabilu potrebno neko vrijeme za promjenu stanja iz jednog u drugo.
- To vrijeme je ključno usko grlo arhitekture modernih računala. Na primjer,
  - brze memorije rade na frekvencijama od preko 1 GHz (i to s raznim trikovima),

  ali, zasad, nisu stigle do 2 GHz.
- Ova razlika u brzini procesora i memorije rješava se na dva načina:
  - paralelizacijom (podjela na više "chipova"),
  - hijerarhijskom građom cijele memorije u računalu.

Memorija — organizacija i izgled

- Slično Turingovom stroju, osnovni elementi memorije su bitovi, ali memorija:
  - nije linearna (poput trake), već je pravokutna (ili kvadratična),
  - i nije beskonačna, nego konačna (pa nije potrebno imati prazni simbol koji znači kraj trake).
- Zašto pravokutna organizacija memorije? Funkcionalno,
  - 1 bit je premala količina informacije za smislenu obradu.

  Zbog toga se bitovi organiziraju u veće cjeline s fiksnim brojem bitova. Svaku takvu cjelinu zovemo riječ memorije, a broj bitova u riječi je duljina riječi.

Memorija — organizacija i izgled (nastavak)

Svrha: Riječ je osnovna cjelina za smisleni podatak.

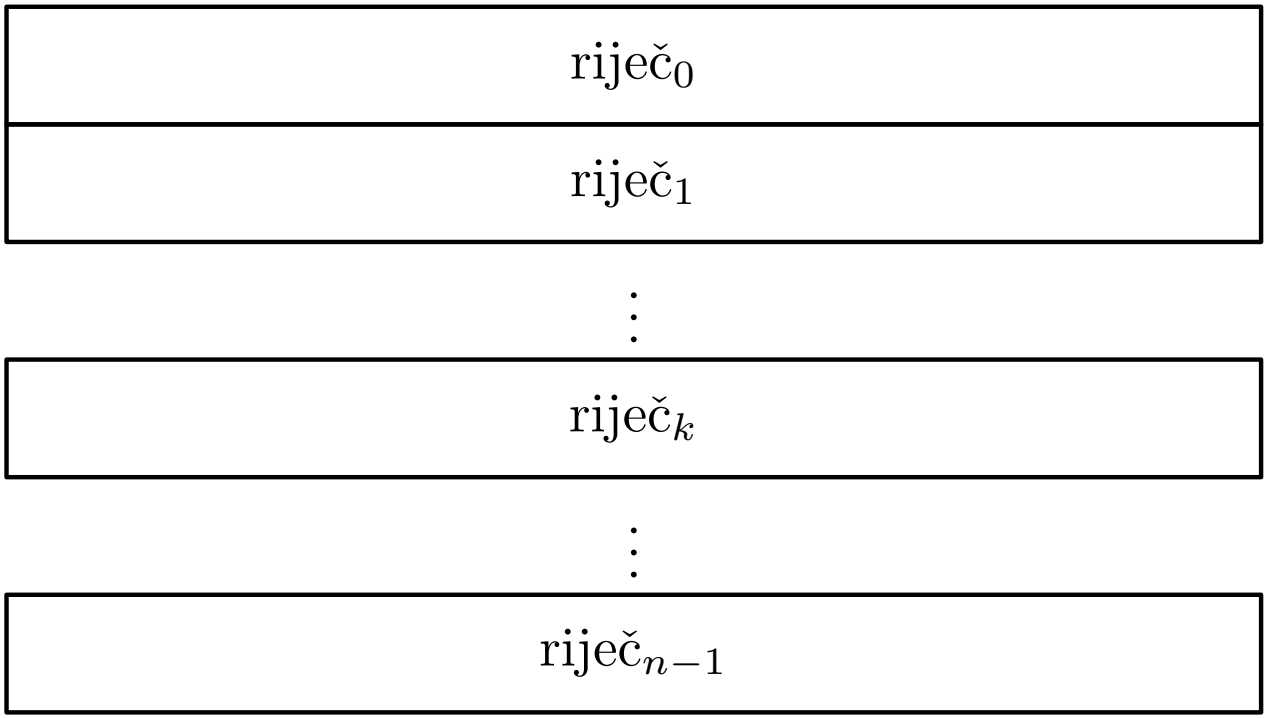
- Takve veće cjeline prikazuju potrebne vrste podataka i na njima kao cjelinama možemo izvršavati pripadne operacije i to brzo na nivou arhitekture računala.
- Koliko bitova čini jednu riječ? Ne postoji točan odgovor, jer to ovisi o tipu podataka koji se prikazuje u arhitekturi računala.
- Pojednostavljeno, riječ je količina bitova predviđena za prikaz cijelih brojeva, ili još bolje, riječ je količina bitova potrebnih za prikaz strojnih instrukcija i adresa.
- Danas, gotovo univerzalno, riječ je količina bitova predviđena za prikaz jednog znaka, tj. riječ = 1 Byte.

Memorija — organizacija i izgled (nastavak)

- Memorija je linearni niz riječi, a svaka riječ ima svoju adresu, tj. poziciju ili mjesto u nizu.
- Slična organizacija vrijedi i za strukturu podataka koju zovemo niz ili polje, tj. to je konačni uređeni niz podatak istog tipa, a pristup pojedinim podacima je moguć preko indeksa u nizu.
- Matematički gledano, niz od n članova je uređena n-torka $x_1, x_2, \ldots, x_n$.
- Razlika obzirom na matematičku definiciju: brojanje pozicije ne počinje s 1 nego s 0, jer pozicija u nizu je adresa koliko je ta riječ "odmaknuta" od početne riječi.

Memorija — organizacija i izgled (nastavak)

- Skica memorije sa n riječi izgleda ovako:



Kažemo da se riječ$_k$ nalazi na k-tom mjestu ili da se nalazi na adresi k.

Memorija (nastavak)

- Da bismo nešto spremili ili pročitali kao sadržaj lokacije u memoriji, moramo imati dvije osnovne instrukcije:
  - spremi podatak na adresu "tu i tu",
  - pročitaj podatak sa adrese "te i te".
- Dakle, pristup podatku ide preko adrese podatka (pozicije podatka u memoriji). Obično još kažemo da adresa "pokazuje" na podatak u memoriji.
- Ključno za razumjevanje rada računala: računalo vidi podatak kao "sadržaj spremljen na određenoj adresi".
- Netrivijalna posljedica: memorijske adrese su također podaci.

Memorija — adresni prostor

- Adresa podatka je ključni dio instrukcije koja nešto radi s podacima.
- Veličina "adresnog" prostora = broj bitova predviđen za spremanje adresa.
- Ako imam m bitova za spremanje adresa, onda mogu prikazati točno 2^m različitih adresa: od 0 do $2^m - 1$.
- To određuje i maksimalnu količinu memorije (više ne mogu adresirati)!
- Adrese se standardno "pišu" u heksadecimalnom sustavu.

Memorija (nastavak)

- Dakle, svaki podatak u memoriji računala ima 2 dijela:
  - adresu — mjesto gdje je spremljen,
  - sadržaj — vrijednost podatka spremljenog na odgovarajućoj adresi, tj. kako se interpretiraju pripadni bitovi.
- Nekad je riječ bila zaista najmanja cjelina koju se moglo direktno adresirati. Recimo:
  - IBM 1130 je imao 16-bitne riječi,
  - mnogi strojevi su imali 32-bitne riječi,
  - Univac 11xx (xx = 06 ili 10) je imao 36-bitne riječi,
  - CDC Cyber su standardno imali 60 ili 64-bitne riječi.

Memorija (nastavak)

- Povijesno, prostor za spremanje 1 znaka teksta, zove se byte. 1 byte = 8 bitova. Znak teksta sprema se u dogovorenom kôdu koji se prikazuje bitovima.
- Oprez — 1 kB = 1 024 bytea, a nije 10^3 bytea, isto tako 1 MB = 1 048 576 bytea, a nije 10^6 bytea.
- Niti to nije baš uvijek bila istina, katkad se za spremanje znaka koristilo 7 ili čak 6 znakova.
- Jasno je da su znakovi jedan od ključnih tipova podataka koje treba spremiti u memoriju.
- Standardi za pisanje znakova pojavili su se istovremeno s prvim računalima (nije lijepo čitati nizove nula i jedinica).

Memorija (nastavak)

- Standardi:
  - EBCDIC — asketskih 6 bitova,
  - ASCII — 7-bitni standard s velikim i malim slovima,
  - 8-bitni ASCII — standard za dodatnih 128 znakova koji su potrebni za odgovarajuće jezike i razlikuje se od jezika do jezika (ISO–nešto character set).

- Nakon toga su se pojavili mikroprocesori čija je memorija bila upravljana po principu 1 riječ = 1 byte, pa je to postala najmanja cjelina koju je procesor mogao adresirati. To znači i da je procesor je bio 8-bitni, a takva je bila i veza procesora i memorije, tzv. sabirnica ili magistrala.

Memorija (nastavak)

- U modernim osobnim računalima (IA-32, AMD64 ili IA-64), byte je i dalje osnovna cjelina koja se može adresirati, međutim stvarna organizacija koristi mnogo dulje riječi:
  - IA-16 — 16-bitna riječ (2 bytea), koristi se za instrukciju + adresu,
  - IA-32 — 32-bitna riječ, vrijedi za većinu današnjih osobnih računala,
  - AMD64, EM64T, x64 — Athlon-64, EM64T Intel, je IA–32, ali su adrese 64-bitne.
  - IA-64 — Itanium, ..., a radi se i na 128-bitnom standardu za velika računala.

Tipovi podataka

- Zasad nismo rekli koji su osnovni tipovi podataka za nas korisnike. Za računanje koristimo brojeve raznih vrsta:
  - nenegativne cijele brojeve (bez predznaka), tj. podskup od $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$,
  - cijele brojeve s predznakom (i negativni uključeni), tj. podskup od $\mathbb{Z}$,
  - brojeve s pomičnim zarezom (engl. floating point), tj. podskup od $\mathbb{R}$.
- Za ulaz–izlaz koristimo: znakove.
- Svi ostali tipovi uglavnom se svode na ove osnovne tipove.

Tipovi podataka (nastavak)

- Veza arhitekture računala i tipova podataka ide tako daleko da se i najjednostavniji tip podataka logički ili Booleov tip, koji ima samo dvije vrijednosti laž ili istina (oznake F/T, $\bot/\top$) prikazuje preko cijelih brojeva i to kao laž $= 0$, istina $= 1$.

- Osim ovih korisničkih tipova trebamo još 2 stvari bitne za rad računala:
  - adresa — to je tip podataka sličan nenegativnim cijelim brojevima, tj. stvarno su im prikazi isti.
  - instrukcije.

- Po von Neumannovom modelu, instrukcije/programi se također pamte u memoriji. Strojne instrukcije se nekako kôdiraju bitovima.

Procesor

- Po von Neumannovom modelu, procesor mora imati bar 2 bitna "radna" dijela:
  - izvršni = aritmetičko logičku jedinicu — naziv dolazi od tipičnih operacija koje ona izvršava nad korisničkim tipovima podataka,
  - upravljački dio — brine se za dohvat instrukcija iz memorije (engl. fetch), njihovu interpretaciju (engl. decode) i za njihovo izvršavanje (engl. execute). Upravljački dio upravlja radom aritmetičko–logičke jedinice prema instrukcijama.

Procesor (nastavak)

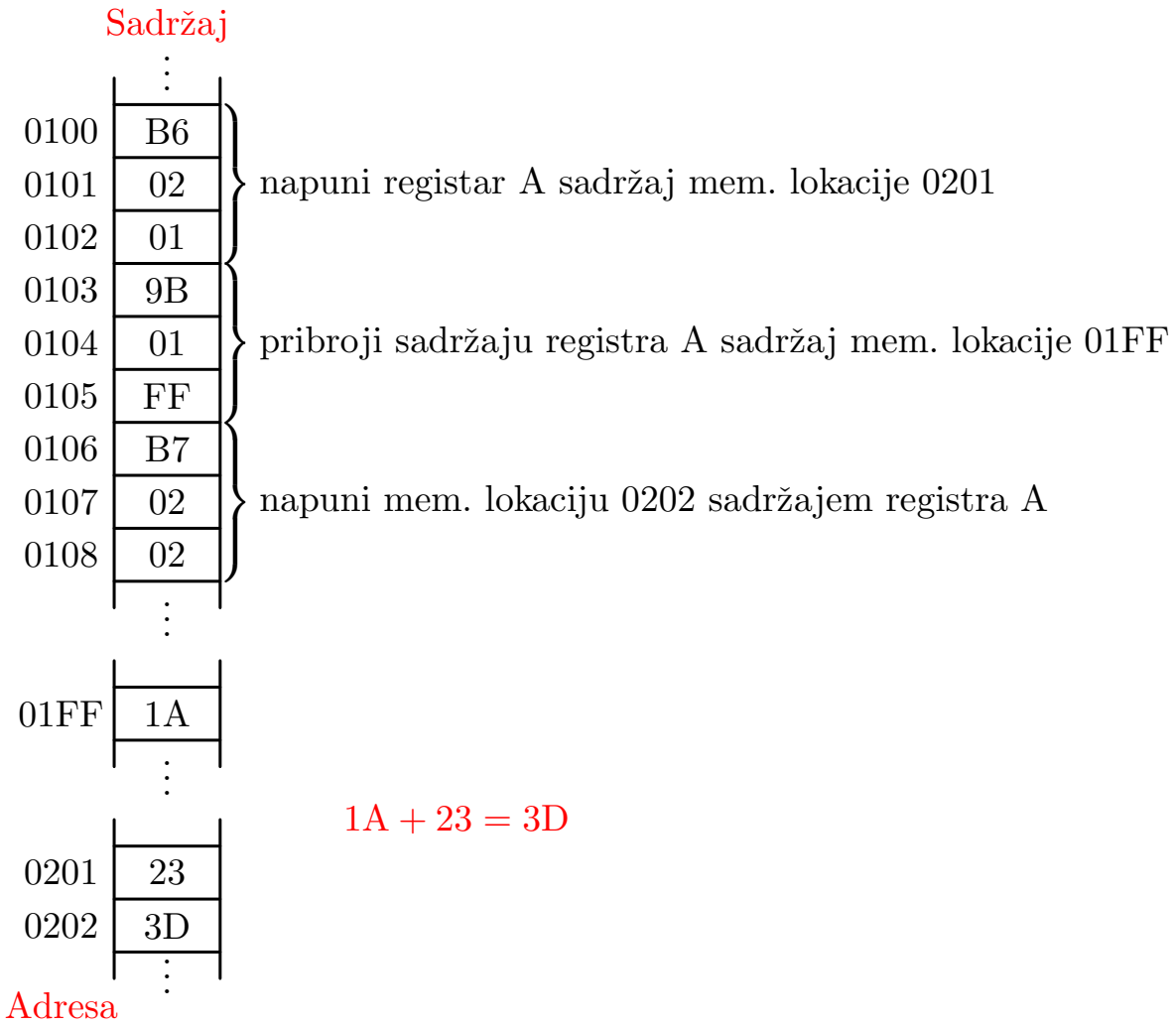
- Osim toga, procesor ima i skup registara. To je radna memorija procesora za spremanje:
  - podataka nad kojima se izvršavaju instrukcije i rezultata tih operacija,
  - instrukcija (odnosno, dijelova instrukcija) koje se izvršavaju.
- Sve operacije u procesoru mogu se napraviti
  - samo na operandima koji su prebačeni iz memorije u registre procesora.

Procesor (nastavak)

- Zašto je organizacija takva? Procesor i memorija su fizički odvojeni i komuniciraju preko "kanala" (magistrala, sabirnica). Za izvršavanje bilo koje instrukcije, prvo treba instrukciju "dovući" iz memorije u procesor. Isto vrijedi i za podatke!
- Osnovne instrukcije za baratanje podacima:
  - LOAD REG, adr — "napuni" registar "REG" s adrese "adr",
  - STORE REG, adr — "spremi" podatak iz registra "REG" na adresu "adr".

Program za zbrajanje dva broja

Program:



Ulazne i izlazne jedinice

Svako računalo, osim memorije i procesora, mora imati još i:

- ulaznu jedinicu — koja podatke iz vanjskog svijeta pretvara u binarni oblik.
- izlaznu jedinicu — koja rezultate iz binarnog oblika pretvara u oblik razumljiv korisniku, ili ih pohranjuje za daljnju obradu.

Shema računala

Shematski, ovako izgledaju osnovni dijelovi računala: