

Programiranje 1

Predavanje 02 - Algoritmi, principi rada i građa računala

Matej Mihelčić

Prirodoslovno-matematički fakultet
Matematički odsjek

24. rujna 2024.



Algoritam je metoda, postupak, pravilo za rješavanje nekog problema ili dostizanje nekog cilja.

- **Postupak** - konačni niz koraka potreban za riješiti problem.
- **Metoda** - za razliku od matematike gdje može označavati i beskonačni *postupak*, ovdje se **mora izvesti u konačnom broju koraka**.

Osnovni cilj - razvoj **efikasnih i točnih** algoritama.

Da bi to mogli moramo znati:

- Koristiti i implementirati postojeće algoritme za rješavanje raznih poznatih problema.
- Analizirati algoritam i odrediti njegovu efikasnost.
- Samostalno osmisliti algoritme za nepoznate probleme.

- Kulinarski recepti.
- Upute za rukovanje raznim strojevima ili uređajima.
- Upute za sastavljanje namještaja, maketa, aviona i brodova od papira i sl.
- Upute za dohvaćanje raznih potvrda (liječničkih, bankovnih, upravnih).
- Zbrajanje, oduzimanje, množenje, dijeljenje brojeva.
- Rastav broja na proste faktore.

Algoritmi se sastoje od **niza koraka**.

- Svaki korak je neka instrukcija ili naredba ili akcija koju treba izvršiti.
- Korak može biti i:
 - Grananje:
*Ako vrijedi uvjet napravi **koraci**₁, inače napravi **koraci**₂*
 - Ponavljanje:
*Ponovi 6 puta sljedeće korake,
Sve dok je ispunjen **uvjet**, ponavljaj akciju/e.*
 - *Skoči na korak i (**Ne koristimo na kolegijima**).*

Instrukcije se uobičajeno pišu **jedna ispod druge**.

Algoritam izvodi zadane operacije nad nekim podacima, u obliku niza koraka, i daje neko rješenje.



Osnovna svojstva algoritma su:

- **ima (ili nema)** ulazne podatke,
- **ima** izlazne podatke,
- završava u **konačnom vremenu**,
- uvijek je **nedvosmisleno definiran**.

Možemo li proizvoljan problem riješiti algoritamski?

NE! Postoje algoritamski nerješivi problemi (Alan Turing je dokazao 1936. godine da ne postoji generalni algoritam koji bi riješio tzv. **Halting problem**).

Možemo li sve algoritamski rješive probleme riješiti efikasno?

Ne znamo. Postoje klase problema, tzv. *NP*-potpuni i *NP*-teški za čije rješavanje trenutno ne postoje efikasni algoritmi (npr. Problem trgovačkog putnika, problem ispunjivosti logičke formule, problem pokrivanja skupa itd.).

Korak algoritma ovisi o *moći* (mogućnosti, elementarnoj operaciji) izvršitelja.

Primjer: Algoritam za kupovinu novog stola.

- ① Naruči stol s web-stranice trgovine namještaja.
- ② U dogovorenom terminu preuzmi paket sa stolom na kućnoj adresi.
- ③ Sastavi stol:
 - 3.1 Otvori paket.
 - 3.2 Izvadi najveću dasku, te 4 vijka broj 37263.
 - 3.3 Izvadi metalne spojnice.
 - 3.4 Izvadi metalne noge stola.
 - 3.5 Zašarafi spojnice za dasku koristeći vijke.
 - ⋮
 - 3.n Postavi stol na noge.
- ④ Smjesti stol na odgovarajuću poziciju u prostoriji.
- ⑤ Stavi potrebne stvari na stol.

Primjer: Algoritam za kupovinu novog stola (drugačija moć izvršitelja).

- 1 Naruči stol i montiranje s web-stranice trgovine namještaja.
- 2 U dogovorenom terminu otvori vrata monteru koji dolazi s paketom koji sadrži stol.
- 3 Monter sastavlja stol.
- 4 Monter smješta stol na odgovarajuću poziciju u prostoriji.
- 5 Stavi potrebne stvari na stol.

Kako izbor programskog jezika utječe na potrebno znanje o principima rada i funkcioniranju računala?

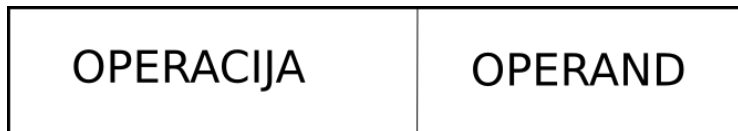
- Programski jezici višeg nivoa sadrže sloj koji omogućava da korisnici/programeri mogu (relativno) učinkovito programirati bez dobrog poznavanja detalja rada i funkcioniranja računala.
- Programski jezici srednjeg i nižeg nivoa omogućavaju programerima direktno upravljanje pojedinim hardverskim komponentama što zahtjeva detaljnije poznavanje rada i funkcioniranja računala.
- Programski jezici najnižeg nivoa (strojni jezik) zahtjeva opsežno poznavanje detalja rada i funkcioniranja računala (uključujući specifični instrukcijski set hardverskih komponenti).

Zašto programski jezik C?

- Dovoljno je visokog nivoa da se algoritmi mogu implementirati relativno **kratkim i čitkim programima**.
- Dovoljno je **niskog nivoa** da **ne skriva detalje od programera**.
- Daje vrlo dobru bazu za razumijevanje *moćnijih* instrukcija drugih programskih jezika i aplikacija općenito.

Računalo je stroj za izvršavanje instrukcija.

Opći oblik instrukcije:



Računalo zna: a) izvršiti osnovne instrukcije, b) raditi s osnovnim podacima.

Računalo ima **ulazni dio** (tipkovnica, miš) i izlazni dio (**monitor, pisač**).

Algoritam **izvršava instrukcije nad podacima koje je učitao, stvara međurezultate** i konačno **rezultat**. Podaci i međurezultati se spremaju u **memoriju** (hard disk, radna memorija, cache). Naredbe se izvršavaju u **izvršnom dijelu računala** (procesor).

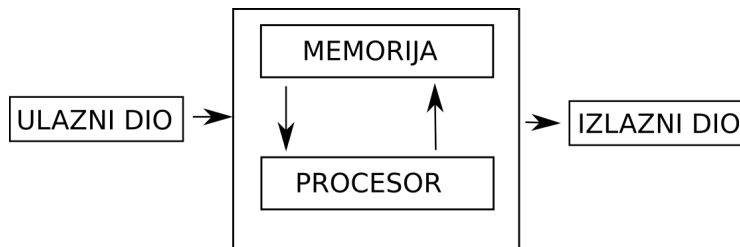
Instrukcije koji čine algoritam:

- Spremljene su na nekom mediju (disk).
- Prilikom pokretanja programa se učitavaju u memoriju.
- Procesor redom izvršava instrukcije iz memorije.

Činjenica da se podaci i instrukcije spremaju u **istu memoriju** je glavna značajka **von Neumannovog modela** računala.

Von Neumannov model računala

Model je prvi puta opisan 1945. u izvještaju o računalu EDVAC. Jednostavniji prethodnik ENIAC, prvo generalno računalo, nije spremalo programe u memoriju (za vrijeme rata korišteno za **računanje balističkih tablica**).



Sastoji se od osnovnih elemenata - **bistabila** (označava 2 stabilna stanja). U računalima su ta dva stanja 1 i 0 stoga svaki bistabil sprema točno 1 bit informacije - jednu binarnu znamenku.

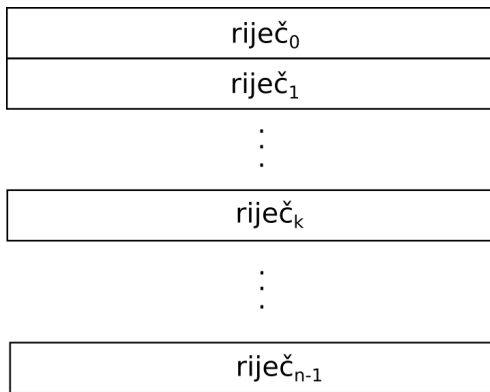
Bitovi se grupiraju u veće cjeline koje se zovu **riječi** (osnovne cjeline smislenih podataka).

Riječ je količina memorije potrebna za spremanje jednog znaka (1 byte = 8 bitova).

Čvrsti disk (HDD) - magnetna glava namagnetizira tanki magnetični sloj postavljen na diskovima od aluminija, stakla ili keramike. **Prednosti:** relativno jeftin, može spremati velike količine informacija, informacije se spremaju i kada nema izvora struje. **Mane:** relativno spor zbog nužnog, fizičkog pomicanja glave radi pisanja/čitanja, mogući kvarovi tehničke prirode.

Radna memorija/SSD (eng. solid state drive) - sastoji se od niza tranzistora i otpornika. Otpornici ili zadržavaju pozitivan napon ili nemaju napona. Tim procesom se pamte informacije. **Prednosti:** puno brži od HDD-a zato što nema potrebe za fizičkim pomacima glave. **Mane:** skuplji, podržava ograničen broj pisanja/brisanja.

Memorija je linearan niz riječi. Riječ ima **adresu**, poziciju u nizu.



Svaki **podatak** ima adresu (mjesto gdje je spremljen) i sadržaj (vrijednost podatka spremljenog na toj adresi).

Memorijske adrese standardno zapisujemo u heksadecimalnom sustavu (kraći zapis - vidi vježbe), npr: `4ABCDEF1`.

Današnja računala za spremanje memorijske adrese obično koriste 64 bita. Moguće je imati maksimalno $2^{64} \approx 18.4 \cdot 10^{18}$ riječi. (byte-ova memorije).

32-bitna računala su mogla adresirati max. 2^{32} byte-ova (4GiB memorije).

Napomena:

- 1KiB (kibibyte) = 1024 byte-ova.
- 1MiB (mebibyte) = $1024 \cdot 1024 = 1048576$ byte-ova.
- 1GiB (gibibyte) = $1024 \cdot 1024 \cdot 1024 = 1073741824$ byte-ova.

Da bismo nešto spremili ili pročitali kao sadržaj lokacije u memoriji, moramo imati dvije osnovne instrukcije:

- Spremi podatak na **zadanu** adresu.
- Pročitaj podatak sa **zadane** adrese.

Netrivijalna posljedica:

- Memorijske adrese su **također podaci**.
- Strojne instrukcije računala **moraju sadržavati memorijske adrese podataka s kojima rade**.

Sastoji se od dva dijela:

- Izvršni dio (**aritmetičko-logička jedinica**)
- **Upravljački dio** - obavlja osnovne poslove vezane uz instrukcije.
 - dohvat instrukcija iz memorije
 - interpretaciju instrukcija
 - izvršavanje instrukcija

Procesor ima i **skup registara** (radna memorija procesora). Služi za spremanje **podataka** i **instrukcija**.

- Omogućavaju izvršavanje logičkih operatora poput NOT, OR, AND nad pojedinim bitovima.

Realizacija aritmetičkih operacija u računalu:

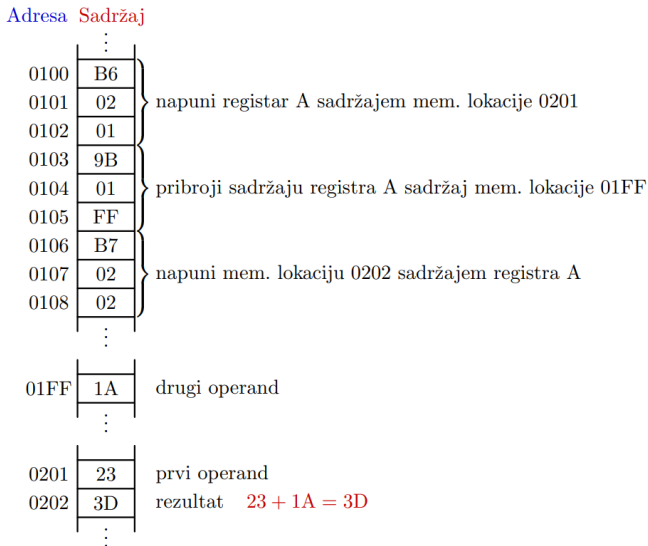
- Bitovi se organiziraju u veće cjeline (npr. 32 bita) i interpretiraju kao brojevi (cijeli, realni).
- Spajanjem više logičkih sklopova u složenije sklopove se mogu implementirati operacije zbrajanja, množenja, dijeljenja itd.

Sve operacije u računalu su realizirane osnovnim logičkim sklopovima koji rade s bitovima iz memorije.

Analogija između logičkih i aritmetičkih operacija nad binarnim znamenkama.

- NOT x (logičko ne), $0 \leftrightarrow 1, 1 - x$
- x AND y (logičko i), $x \cdot y$
- x OR y (logičko ili), $x + y - x \cdot y$

Zapis u memoriji: programski kod i podaci



Schema računala

