Uvod u računarstvo
3. predavanje

Saša Singer

singer@math.hr

PMF – Matematički odjel, Zagreb

Sadržaj predavanja

- Turingov, apstraktni stroj za izvođnje algoritma.
  - glavni dijelovi Turingovog stroja: traka, glava, stanja stroja, program.
- Građa računala:
  - memorija (bistabil, bit, riječ),
  - procesor (aritmetičko–logička jedinica, upravljačka jedinica),
  - ulazna jedinica,
  - izlazna jedinica.

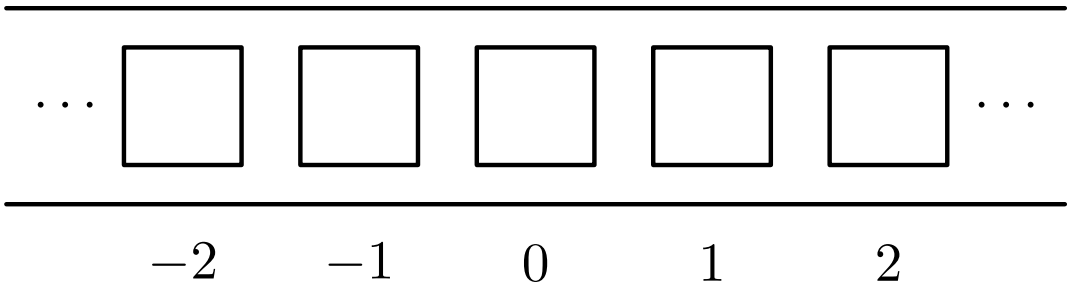
Turingov stroj

- Prije prelaska na detalje građe procesora i memorije, zanimljivo je pogledati kako izgleda matematički (apstraktni) stroj za izvršavanje algoritma, tzv. Turingov stroj.
- Turingov stroj je kao ideja stariji od von Neumannovoga. Nastao je krajem dvadesetih i početkom tridesetih godina prošlog stoljeća.
- Autor Alan Turing logičar, koji nije bio samo teoretičar, već je projektirao specijalna računala koja su Britanci u Bletchley Parku koristili za razbijanje šifri njemačkih strojeva za šifriranje (Enigma).

Turingov stroj – traka

- Kako izgleda Turingov stroj? On se sastoji od nekoliko bitnih dijelova.

(a) Turingov stroj ima dvostrano beskonačnu traku koja sadrži polja (“kvadratiće”) numerirane cijelim brojevima:



- Svako polje može sadržavati jedan znak iz nekog skupa znakova koji stroj prepoznaje, tj. kojeg zna “pročitati” i “napisati”.

Turingov stroj – traka (nastavak)

- Jednostavnosti radi, uzmimo da se taj skup znakova sastoji iz samo 3 simbola: ‘0’, ‘1’ i ‘ ’ (“praznina”).
- Prva 2 znaka su binarne znamenke i služe za zapis “korisnih” informacija na traci, a praznina služi kao oznaka kraja podataka. (Konačnost niza binarnih znamenki osigurana prazninom!)
- Mogli smo uzeti i veći skup znakova, ali manji ne smijemo! Razlog: kôdiranje ulaza mora biti razumno kratko.

Primjer kôdiranja

- Pogledajmo kako u ‘0’, ‘1’ alfabetu kodiramo nenegativne brojeve. Ako je $n \in \mathbb{N}$, onda ćemo ga u takvom alfabetu kodirati binarnim znamenkama $0 \mapsto$ ‘0’ i $1 \mapsto$ ‘1’. Broj n prikazat ćemo u bazi $b = 2$ kao:

$$n = a_k 2^k + \cdots + a_1 \cdot 2 + a_0, \quad a_i \in \{0, 1\}, \quad a_k > 0.$$

  U stroju će to biti prikazano kao niz znamenki:

$$a_k,\ a_{k-1},\ \ldots,\ a_1,\ a_0.$$

- Uz to, moramo se dogovoriti da je zapis nule $a_0 = 0$ duljine jedne znamenke.

Primjer kôdiranja (nastavak)

- Pogledajmo koliko nam je znakova z potrebno za kodiranje broja n. Očito je

$$2^k \leq n < 2^{k+1}.$$

Logaritmiranjem dobivamo $k \leq \log_2 n < k+1$, pa je $\lfloor \log_2 n \rfloor = k$, što znači da je

$$\lfloor \log_2 n \rfloor \leq \log_2 n < \lfloor \log_2 n \rfloor + 1.$$

Dakle, za kodiranje nam je potrebno:

$$z = \begin{cases} \lfloor \log_2 n \rfloor + 1, & n > 0, \\ 1, & n = 0 \end{cases}$$

znamenki.

Primjer kôdiranja (nastavak)

- Što ako uzmem manji alfabet, koji se sastoji samo od ‘0’ i ‘ ’, uz dogovor da praznina opet služi za oznaku kraja ulaza.
- Tada se svaki $n \in \mathbb{N}$ može zapisati korištenjem n znakova 0, pa je duljine zapisa linearna u n!
- Javlja se još jedan problem, kako zapisati broj 0. Očiti zapis 0 nije moguće koristiti jer smo ga “potrošili” na zapis broja 1.
- Zaključak: nije dobro koristiti alfabet sa samo 2 znaka, jer je duljina zapisa linearna, a ne više logaritamska u n, što može drastično utjecati na složenost algoritama.

Turingov stroj – glava

- Iz svega što smo dosad rekli, očito je da traka služi kao memorija Turingovog stroja.

(b) Traka Turingovog stroja ima glavu koja može napraviti sljedeće operacije s trakom:

- pročitati jedan znak s trake (s polja koje se nalazi "ispod" glave),
- napisati jedan znak na traku (u polje "ispod" glave),
- pomaknuti se, relativno obzirom na trenutnu poziciju za jedno mjesto (polje) u oba smjera, tj. može napraviti pomak $+1$ (značenje: pomakni se jedno mjesto nadesno) i -1 (značenje: pomakni se jedno mjesto nalijevo).

Turingov stroj – programski dio

- Pisanje, čitanje i pomaci glave Turingovog stroja pokazuju da ona služi kao kontrolni mehanizam Turingovog stroja.

(c) "Programski dio" Turingovog stroja sastoji se od konačnog niza stanja u kojima se on može nalaziti. U svakom trenutku stroj se nalazi u točno jednom od mogućih stanja.

Moguća stanja su podijeljena u 3 "vrste":

- "regularna stanja": ili "međustanja" ili radna stanja, zovemo ih $q_1, \ldots, q_s$,
- "početno stanje": zovemo ga q_0,
- "završno stanje": zovemo ga q_f.

Turingov stroj – programski modul

- Katkad se uzima da ima više završnih stanja, ali tada ona odgovaraju rezultatu algoritma. Na primjer, ako algoritam daje odgovor da/ne na neko pitanje, onda stroj ima 2 završna stanja q_y ako je odgovor da i q_n ako je odgovor ne. To nije jako bitno, jer odgovor možemo i zapisati na traku!

(d) Program ili programski modul kojeg možemo ugraditi u stroj. On upravlja strojem i vodi ga kroz korake pojedinog algoritma.

Kako radi programski modul? Ovisi o trenutnom stanju stroja i znaku koji glava učita s trake. Tada stroj prijeđe u neko drugo stanje, napiše neki znak na traku i miče glavu jedno mjesto udesno ili ulijevo.

Turingov stroj – programski modul (precizno)

- Precizniji opis kako radi programski modul. Uzmimo da je stroj u nekom stanju q, koje nije završno, $q \neq q_f$ i da je glava u tom trenutku učitala "simbol" s trake. Na osnovu para

$$(q, \text{simbol})$$

programski modul odlučuje 3 stvari:

(i) koje je sljedeće stanje q' u koje prelazi stroj,

(ii) koji će znak napisati u polje ispod glave (taj znak zovemo "novi simbol"),

(iii) miče li se traka jedno polje nadesno ili jedno polje nalijevo (pomak za $+1$ ili -1).

Turingov stroj – programski modul (precizno)

- Dakle, jedan programski korak je veza

$$(q, \text{simbol}) \quad \text{u} \quad (q', \text{novi simbol}, \text{pomak}).$$

  Ako je ova veza funkcija, tj. svakom paru (q, simbol) pridruži točno jednu trojku $(q', \text{novi simbol}, \text{pomak})$, onda je stroj deterministički. Ali to ne mora biti tako!

- Kad stroj dođe u završno stanje q_f, onda se računanje prekida.

- Na početku stroj je u stanju q_0, onda, a glava se nalazi nad poljem s brojem 1.

- Ovim smo, zapravo, napravili sve elemente za formalnu definiciju Turingovog stroja.

Građa računala – memorija

- 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. Zbog toga se ta stanja uobičajeno i označavaju binarnim znamenkama 0 i 1.

Građa računala – memorija (nastavak)

- Nekad, u doba ranih računala (1960-tih) bistabil se realizirao pomoću feritnih jezgrica. Feritne jezgrice sastojale su se od sitnih 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 (opet imaju 2 stanja), ima struje – nema struje.

- Osnovne logičke operacije (ne, i, ili) operiraju kao aritmetičke na 0, 1 (promjena predznaka, zbrajanje, množenje). Logičkim sklopovima mogu se realizirati osnovne logičke operacije na pojedinim bitovima!

Građa računala – memorija (nastavak)

- Kad bitove organiziramo u veće cjeline (na pr. dogovor prikaza prirodnih brojeva binarnim znamenkama), pomoću takvih logičkih sklopova mogu se realizirati i osnovne aritmetičke operacije na brojevima (zbrajač).

- Čisto tehnički tu su 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 je aritmetika binarna.
  - Brzina sjetlosti je trenutno fundamentalno ograničenje brzine računala (minijaturizacija: 130 nm, 90 nm, . . . tehnologije).

Građa računala – memorija (nastavak)

- Brzinom upravljanja operacija diktira brzina svjetlosti (puštanje struje kroz vodiče, a onda sve ovisi o tome jesu li prekidači otvoreni ili ne). Logički sklopovi su još relativno brzi.
- Kod memorija, situacija je kompliciranija, jer je bistabilu potrebno neko vrijeme za promjenu stanja s jednog u drugo. To vrijeme je ključno usko grlo arhitekture modernih računala.
- Slično Turingovom stroju, osnovni elementi su bitovi, ali memorija nije linearna, već kvadratična i nije beskonačna, pa nije potrebno imati prazni simbol koji znači kraj trake.

Građa računala – memorija (nastavak)

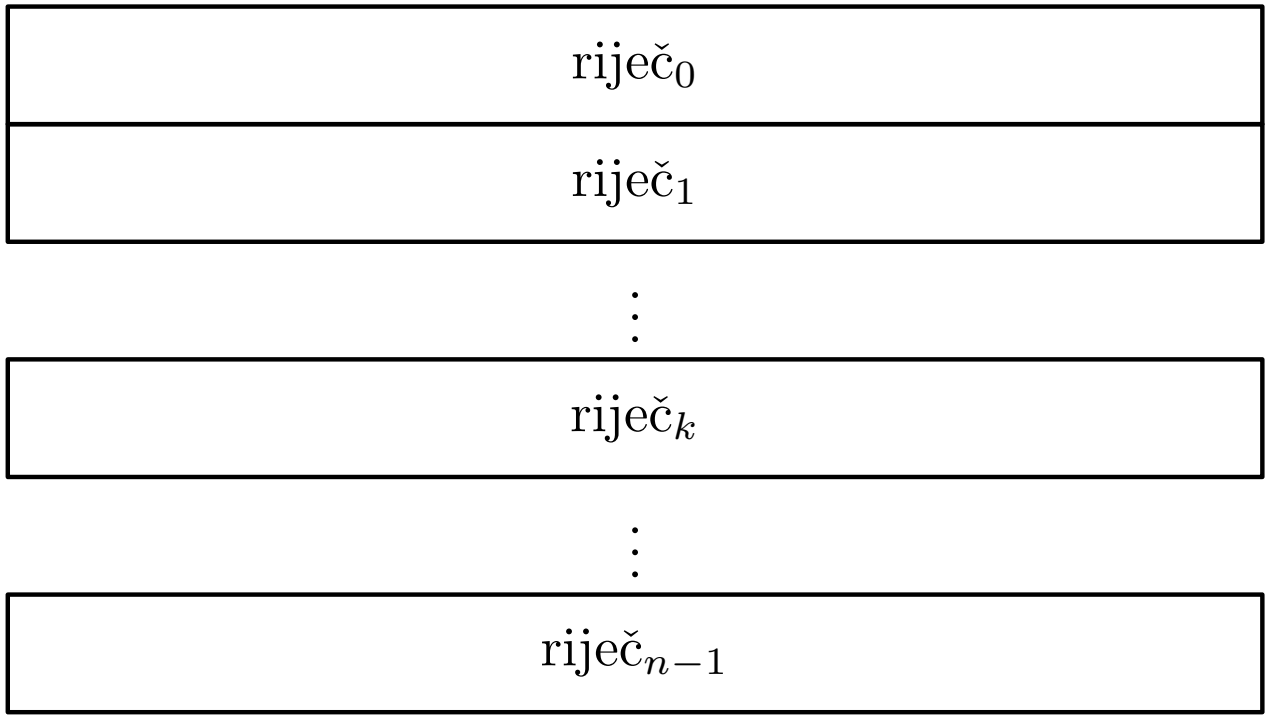
- Zašto kvadratična organizacija memorije? Funkcionalno – bit je premala količina informacije za smislenu obradu. Zbog toga se bitovi organiziraju u veće cjeline koje prikazuju potrebne vrste podataka i na kojima kao cjelinama možemo izvršavati pripadne operacije i to brzo na nivou arhitekture računala.
- Riječ je osnovna cjelina za smisleni podatak. 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.

Građa računala – memorija (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 0 nego s 1, jer pozicija u nizu je adresa koliko je ta riječ "odmaknuta" od početne riječi.

Građa računala – memorija (nastavak)

- Skica memorije sa n riječi izgleda ovako:



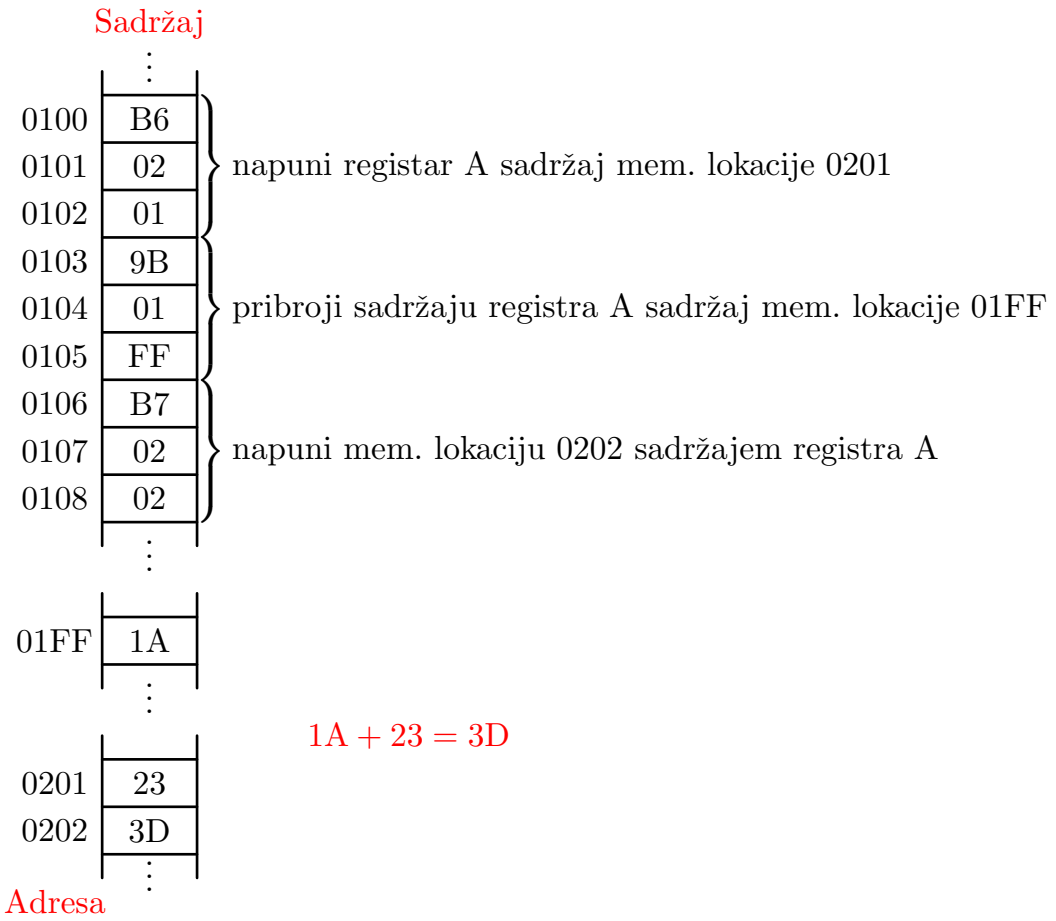
Kažemo da je riječ_k nalazi na k-tom mjestu ili da se nalazi na adresi k.

Građa računala – 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.

Program

- Program:



Građa računala – 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.

Građa računala – 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 je 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).

Građa računala – 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.

Građa računala – memorija (nastavak)

- U modernim osobnim računalima (IA-32 ili IA-64), byte je i dalje osnovna cjelina koja se može adresrati, međutim stvarna organizacija koristi mnogo dulje riječi:
  - IA-16 – 16-bitna riječ (2 bytea) koristi se za instrukciju + adresu,
  - IA-32 – 32-bitna riječ, (imaju je većina današnjih osobnih računala),
  - IA-64 – Athlon-64, Itanium, EMT-64 Intel . . . , 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.

Građa računala – procesor

- Po von Neumannovom modelu procesor mora imati bar 2 bitna 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.

Građa računala – procesor (nastavak)

- Zašto je organizacija takva? Procesor i memorija su fizički odvojeni i komuniciraju preko “kanala”. Za izvršavanje bilo koje instrukcije, prvo treba instrukciju “dovući” iz memorije u procesor.
- Sve operacije se u procesoru mogu naprviti samo na operandima koji su također prebačeni iz memorije u procesor u tzv. registre.
- 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”.

Građa računala – ostalo

- 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: