Uvod u računarstvo
3. predavanje

Saša Singer

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

PMF – Matematički odjel, Zagreb

Sadržaj predavanja

- Uvod u građu i funkcioniranje računala:
  - Turingov stroj.
- Građa računala:
  - memorija,
  - procesor,
  - ulazna jedinica,
  - izlazna jedinica.
- Stvarni "izgled" računala:
  - primjer matične ploče, blok–dijagram,
  - hijerarhijska struktura memorije (cache).
  - registri modernog procesora (IA–32),

“Matematički” model računala

Sadržaj

- "Matematički" model računala — Turingov stroj:
  - ideja i važnost.
- Glavni dijelovi Turingovog stroja:
  - traka,
  - glava,
  - stanja stroja,
  - program.

Uvod — modeli računala

Prošli puta smo ukratko opisali von Neumannov model računala. Osnovna stvar u tom modelu:

- podaci i instrukcije (algoritam) spremljeni su u istoj memoriji.

Prednost: efikasna podloga za realizaciju računala opće namjene (izvršavanje raznih algoritama).

Sad bismo trebali opisati kako izgledaju osnovni dijelovi takvog računala — memorija i procesor (to je osnova za pisanje algoritama u nekom programskom jeziku).

Prije toga, zanimljivo je pogledati kako izgleda

- matematički model računala, tzv. Turingov stroj.

(Poslije se nećemo vraćati na to.)

Turingov stroj

Turingov stroj je

- matematički (apstraktni) stroj za izvršavanje algoritma.

Važnost u matematici (tzv. Churchova teza):

- sve što se uopće "može algoritamski izračunati" (bilo u matematici, bilo u praksi), može se realizirati Turingovim strojem.

Drugim riječima:

- Turingov stroj je univerzalni model računala ("nema jačeg" stroja).

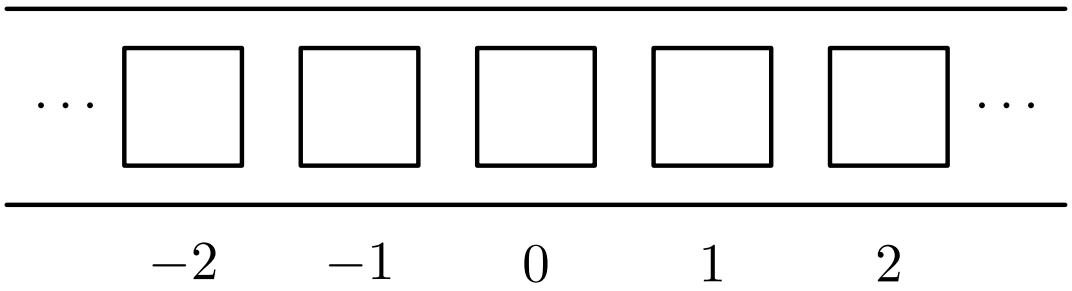
Turingov stroj — malo povijesti

- 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).
- Jedna od najvećih nagrada u računarstvu je Turingova nagrada. Dodjeljuje ju ACM.
  - Zadnji dobitnik (2005): Peter Naur, autor jezika Algol 60.

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 zapisa podataka.
- Uočiti: praznina osigurava konačnost zapisa na traci ("ulaznog" niza binarnih znamenki)!
- 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 cijele 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:

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

  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, za $n > 0$. 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 uzmemo 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) nadesno (pomak $+1$) ili za jedno mjesto nalijevo (pomak -1).

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 stroj može nalaziti. U svakom trenutku stroj se nalazi u točno jednom od mogućih stanja.

Moguća stanja su podijeljena u 3 "vrste":

- "regularna stanja" ("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 — završna stanja

- Katkad se uzima da stroj ima više završnih stanja — koja odgovaraju raznim mogućim rezultatima 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.

Međutim, to nije jako bitno. Zapisivanjem odgovora na traku uvijek možemo postići da stroj ima jedno završno stanje.

Turingov stroj — programski modul

(d) Program ili programski modul koji stvarno 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
- pomiče glavu jedno mjesto udesno ili ulijevo.

Dakle, različita stanja stroja su neka vrsta "programske" memorije stroja.

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 sljedeće 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) pomiče 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

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

  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 (završava).

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

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

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

- 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.

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!

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, 65 nm, ... tehnologije).

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.

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.

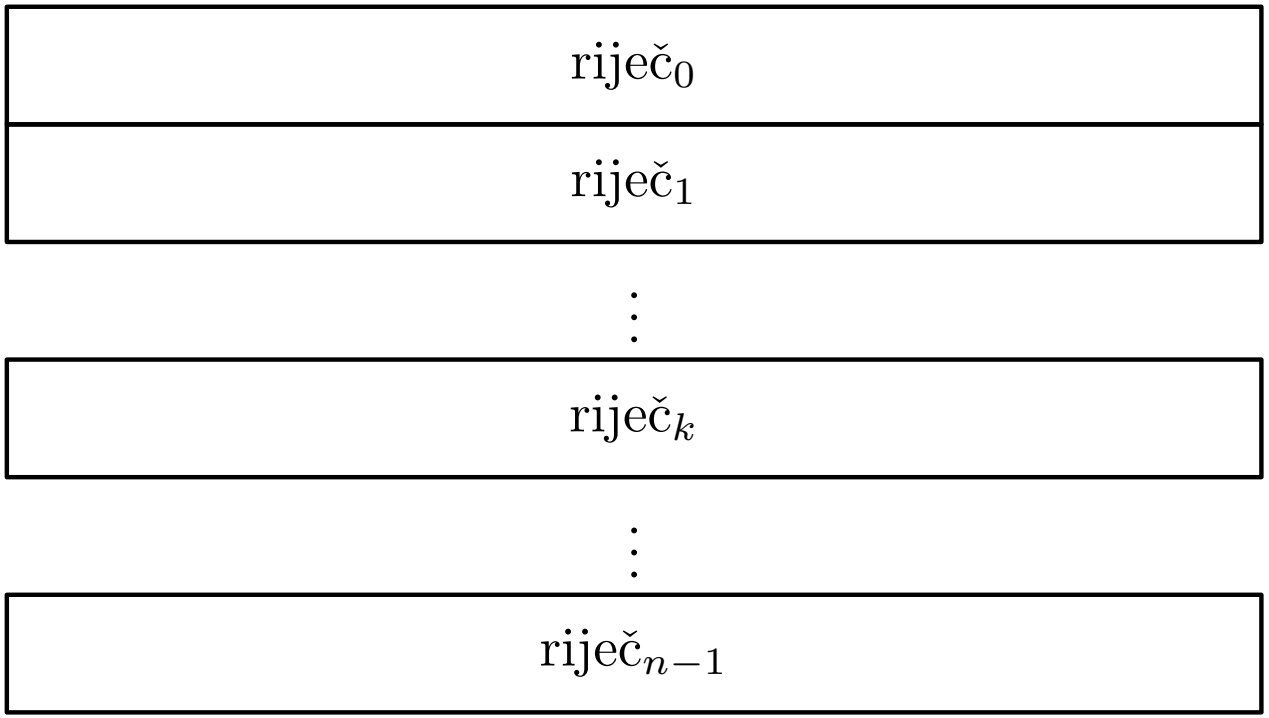
- Pojednostavljeno, riječ je količina bitova predviđena za prikaz cijelih brojeva, ili još bolje, riječ je količina bitova potrebnih za prikaz strojnih instrukcija i adresa.

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.

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.

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 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).

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, (imaju je većina 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

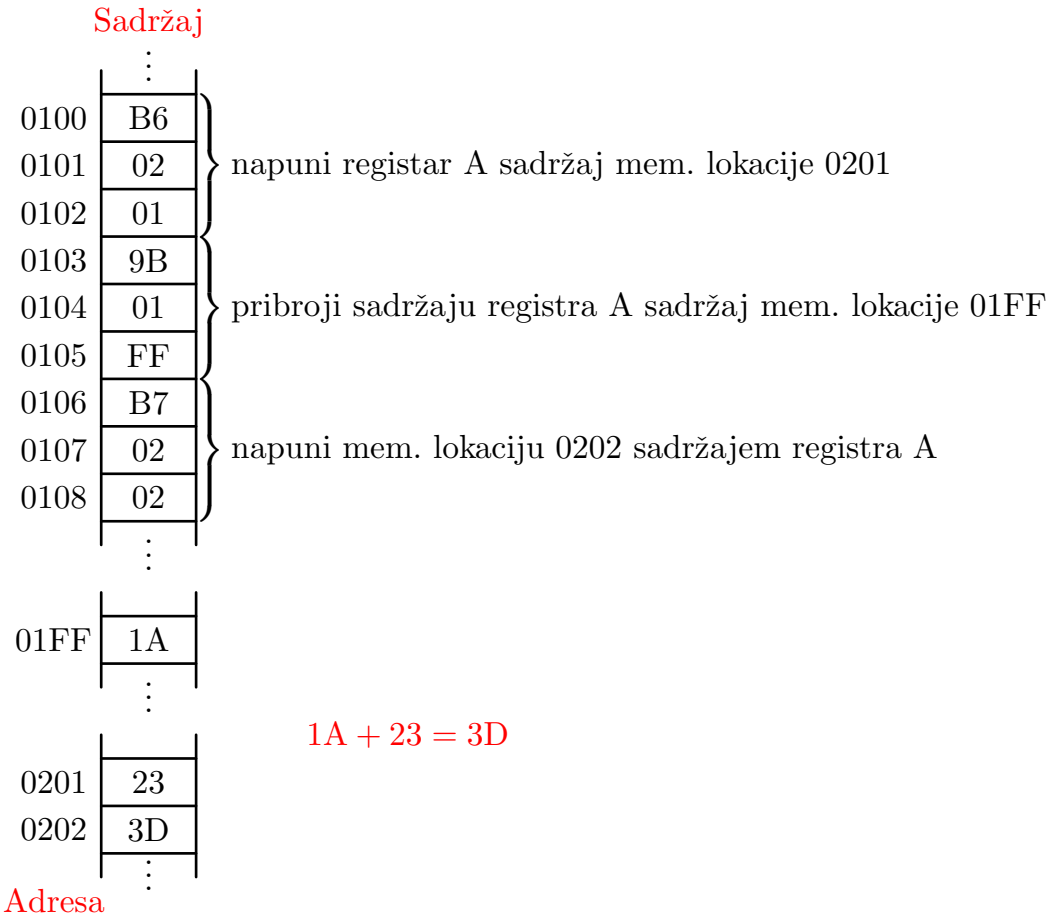
- Po von Neumannovom modelu procesor mora imati bar 2 bitna dijela:
  - izvršni = aritmetičko logičku jedinicu — naziv dolazi od tipičnih operacija koje ona izvršava nad korisničkim tipovima podataka,
  - upravljački dio — brine se za dohvat instrukcija iz memorije (engl. fetch), njihovu interpretaciju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

Procesor (nastavak)

- 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.
- Sve operacije se u procesoru mogu napraviti 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".

Program

- Program:

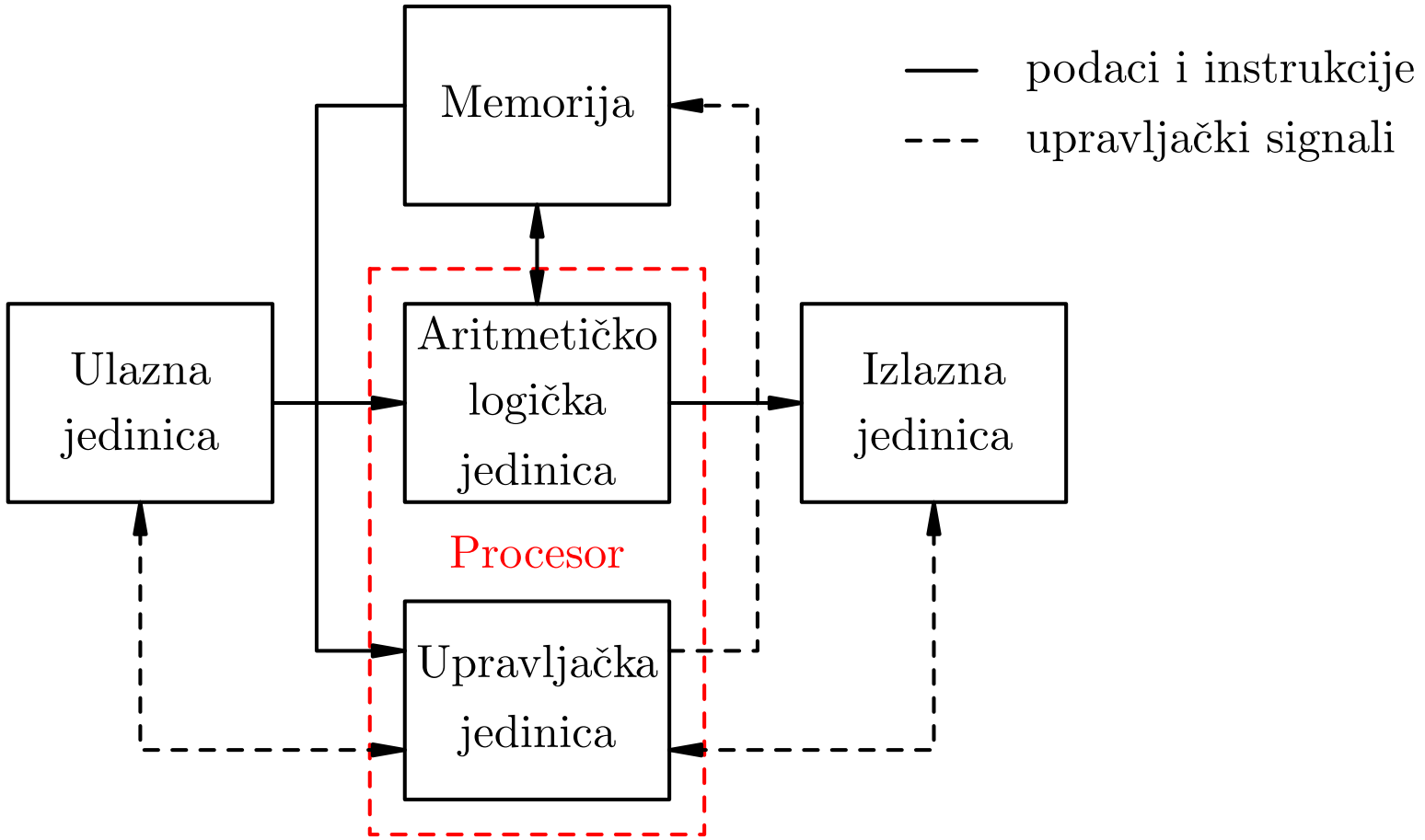


Ulazne i izlazne jedinice

- 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:



Stvarni "izgled" računala

Sadržaj

- Stvarni "izgled" računala:
  - primjer matične ploče, blok–dijagram,
  - hijerarhijska struktura memorije (cache).
  - registri modernog procesora (IA–32),

Izgled matične ploče računala

Moderna "kućna" računala, naravno, imaju sve standardne dijelove računala.

- Međutim, zbog "multimedijalne" namjene, ta računala imaju mogućnost priključivanja velikog broja raznih uređaja ("ulaz–izlaz").
- Gomila toga je već ugrađena na modernim tzv. matičnim pločama (engl. motherboard).
- Procesor zauzima relativno "mali" dio površine (ili prostora), a najuočljiviji dio na njemu (nakon ugradnje) je hladnjak.
- Utori za memorijske "chipove", također, ne zauzimaju previše prostora.

Izgled matične ploče računala (nastavak)

Zbog bitno različite brzine pojedinih dijelova računala, postoje još dva bitna "chipa" koji povezuju razne dijelove i kontroliraju komunikaciju — prijenos podataka između njih. To su:

- Tzv. "northbridge" (sjeverni most), koji veže procesor s "bržim" dijelovima računala. Standardni brzi dijelovi su:
  - memorija,
  - grafika (grafička kartica).
- Tzv. southbridge (južni most), na kojem "visi" većina ostalih "sporijih" dijelova ili vanjskih uređaja.

Izgled matične ploče računala (nastavak)

- Tipični uređaji vezani na southbridge su:
  - diskovi (koji mogu biti i na dodatnim kontrolerima),
  - diskete,
  - komunikacijski portovi,
  - port za pisač (printer),
  - USB (Universal Serial Bus) portovi,
  - tzv. Firewire (IEEE 1394a, b) portovi,
  - mrežni kontroleri,
  - audio kontroleri,
  - dodatne kartice u utorima na ploči (modem), itd.

Izgled matične ploče računala (nastavak)

Veze između pojedinih dijelova idu tzv. "magistralama" ili "sabirnicama" (engl. bus, koji nije autobus).

- Ima nekoliko magistrala, raznih brzina.
- Na istoj magistrali može biti više uređaja, i oni su, uglavnom, podjednakih brzina.

Uočite hijerarhijsku organizaciju komunikacije pojedinih dijelova:

- najsporiji su vezani na ponešto brže,
- ovi na još brže,
- i tako redom, do najbržeg — procesora.

Ova hijerarhija je ključna za efikasnu komunikaciju!

Hijerarhijska struktura memorije

Nažalost, ova hijerahija komunikacije nije dovoljna za efikasnost modernog računala. Grubo govoreći, fali joj vrh, koji se ne vidi dobro na izgledu matične ploče.

- Pravo i najgore usko grlo u prijenosu podataka je komunikacija između procesora i memorije.

Gdje je problem?

Podsjetimo: bilo koje operacije nad bilo kojim podacima možemo napraviti samo u procesoru — preciznije, u registrima procesora. To znači da

- prije same operacije, podatak moramo "dovući" iz obične memorije u neki registar procesora.

Baš to je sporo!

Hijerarhijska struktura memorije (nastavak)

Na primjer, ako procesor radi na 3.6 GHz, a memorija na 533 MHz, onda će

- prijenos podatka u registar trajati okruglo 6 puta dulje od operacije na njemu.

Nažalost, isti tehnološki problem se javlja kod svih modernijih računala.

- Obična radna memorija je bitno sporija od procesora.

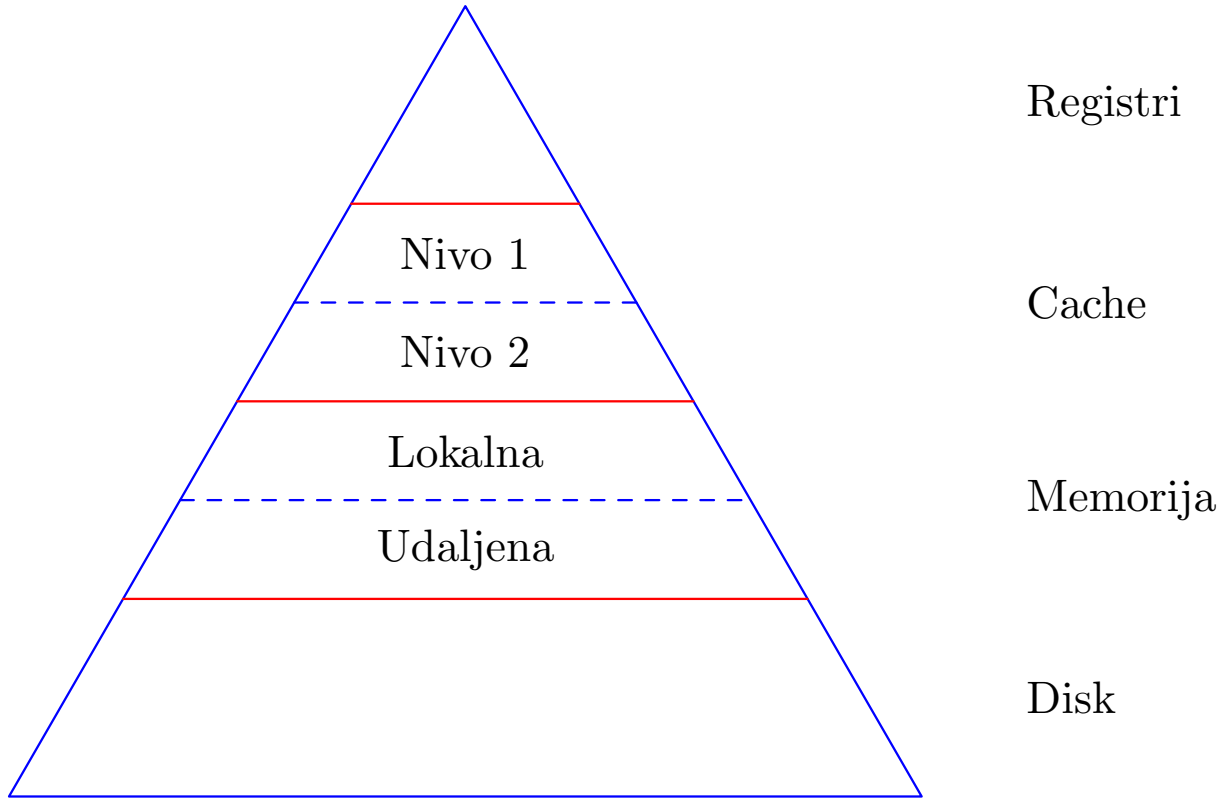
Kako se to izbjegava, ili, barem ublažava?

- Dodatnom hijerhijskom strukturom memorije, između obične radne memorije (RAM) i registara procesora.

Ta "dodatna" memorija se tradicionalno zove cache.

Hijerarhijska struktura memorije (nastavak)

Globalna struktura memorije u računalu ima oblik:



Cache memorija

Dakle, cache je mala i brza "lokalna" memorija — bliža procesoru od obične memorije (RAM). Gdje se nalazi?

- Obično, na samom procesorskom chipu, da bude što bliže registrima.

Nadalje, i taj cache je hijerarhijski organiziran. U modernim procesorima postoji nekoliko nivoa (razina) cache memorije.

- L1 cache za podatke i instrukcije — najbrži, veličina (trenutno) u KB.
- L2 cache za podatke — nešto sporiji, danas obično na frekvenciji procesora, veličina već u MB.
- Katkad postoji i treća razina — L3 cache.

Cache memorija (nastavak)

Na primjer, moj "notebook" ima Intel Pentium 4–M procesor koji na sebi ima (bez pretjeranih tehničkih detalja):

- L1 cache za podatke — 8 KByte-a,
- L1 cache za instrukcije — 12 K tzv. mikro–operacija,
- L2 cache — 512 KByte-a, na frekvenciji procesora.

Ovo su tipični omjeri veličina za Intelove procesore.

Za usporedbu, na AMDovim procesorima omjeri su bitno drugačiji:

- L1 cache je veći,
- L2 cache nešto manji (i, katkad, sporiji).

(Ne ulazimo u to što je bolje!)

Cache memorija (nastavak)

Kako (ugrubo) radi cache?

Kad računalo (tj. njegov operacijski sustav) izvršava neki naš program, onda

- uglavnom, imamo kontrolu sadržaja obične memorije koju taj naš program koristi za podatke i naredbe.

Za razliku od toga,

- nemamo nikakvu izravnu kontrolu nad sadržajem cache memorije.

Naime, cache nije izmišljen zato da bude mala, brža kopija obične memorije i tako ubrza ukupni rad računala.

Cache memorija (nastavak)

Puno je efikasnije da

- cache sadrži podatke koji se češće koriste.

Isto vrijedi i za instrukcije. Dakle, osnovna ideja je:

- "Skrati put do onog što ti često treba".

Naravno, ključna stvar za efikasnost je:

- Što znači "češće" korištenje nekog podatka ili instrukcije?

Dobra globalna ili prosječna efikasnost postiže se samo ako se to odnosi na sve što računalo izvršava u nekom trenutku, tj. na sve pokrenute korisničke programe i dijelove operacijskog sustava.

Cache memorija (nastavak)

U tom svjetlu, kad malo bolje razmislite,

- zaista bi bilo nepraktično da svaki programer određuje što i kada treba ići u koju cache memoriju,

jer prosječna efikasnost nipošto ne ovisi samo o njegovom programu. Zato nema posebnih naredbi za

- učitavanje podataka u cache, ili
- pisanje podataka iz cachea u običnu memoriju.

Umjesto toga, sadržajem cachea upravljaju posebni cache kontroleri, koji

- raznim tehnikama "asocijacije" na više načina povezuju nedavno korištene podatke i instrukcije s onima koje tek treba iskoristiti i izvršiti.

Cache memorija (nastavak)

Bez puno tehničkih detalja, ova asocijacija se realizira otprilike ovako:

- Za svaki sadržaj (podatak ili instrukciju) u cacheu, dodatno se pamti i adresa (iz RAM-a), s koje je taj sadržaj stigao.
- Ako procesor (uskoro) zatraži sadržaj s te adrese, on se "čita" iz cachea (tj. ne treba po njega ići u RAM).
- Po istom sistemu, u cacheu se pamte i stvari koje se "pišu" u običnu memoriju (na putu u RAM).
- Tada se iz cachea brišu podaci koji su najstariji, odnosno, najmanje korišteni (u zadnje vrijeme, otkad su u cacheu).

Cache memorija (nastavak)

Dakle, sadržaj cachea se stalno obnavlja, tako da

- cache čuva najčešće nedavno korištene sadržaje koji bi uskoro mogli trebati.

Iskustvo pokazuje da se isti sadržaji vrlo često koriste više puta, pa se ovo isplati.

Očiti primjer:

- instrukcije u petljama se ponavljaju puno puta!

Ne zaboravimo da je upravo to svrha programiranja i osnovna korist računala.

Cache memorija (nastavak)

Malo kompliciranije je s podacima.

- Ako naš algoritam ne koristi iste podatke puno puta, onda nam cache neće ubrzati postupak.
- U suprotnom, isplati se preurediti algoritam tako da iste podatke koristi puno puta, ali u kratkom vremenskom razmaku — da ne "izlete" iz cachea. (To je neizravna kontrola nad sadržajem cachea.)

Primjeri iz linearne algebre:

- zbrajanje matrica, $C = A + B$ — cache ne pomaže puno;
- množenje matrica, $C = C + A * B$ — dobro korištenje cachea može ubrzati množenje matrica i za 5 puta.