

# *Uvod u računarstvo*

## *4. predavanje*

Saša Singer

`singer@math.hr`

PMF – Matematički odjel, Zagreb

# Sadržaj predavanja

- Građa računala u praksi:
  - primjer matične ploče, blok–dijagram,
  - hijerarhijska struktura memorije (cache).
  - registri modernog procesora (IA–32),
- Osnovni tipovi podataka u računalu (1. dio):
  - višekratnici byte-a (bez strukture),
  - cijeli brojevi bez predznaka, adrese — aritmetika,
  - cijeli brojevi s predznakom — aritmetika.

(Tzv. “realni brojevi” — sljedeći put.)

## 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 hijerarhija 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?

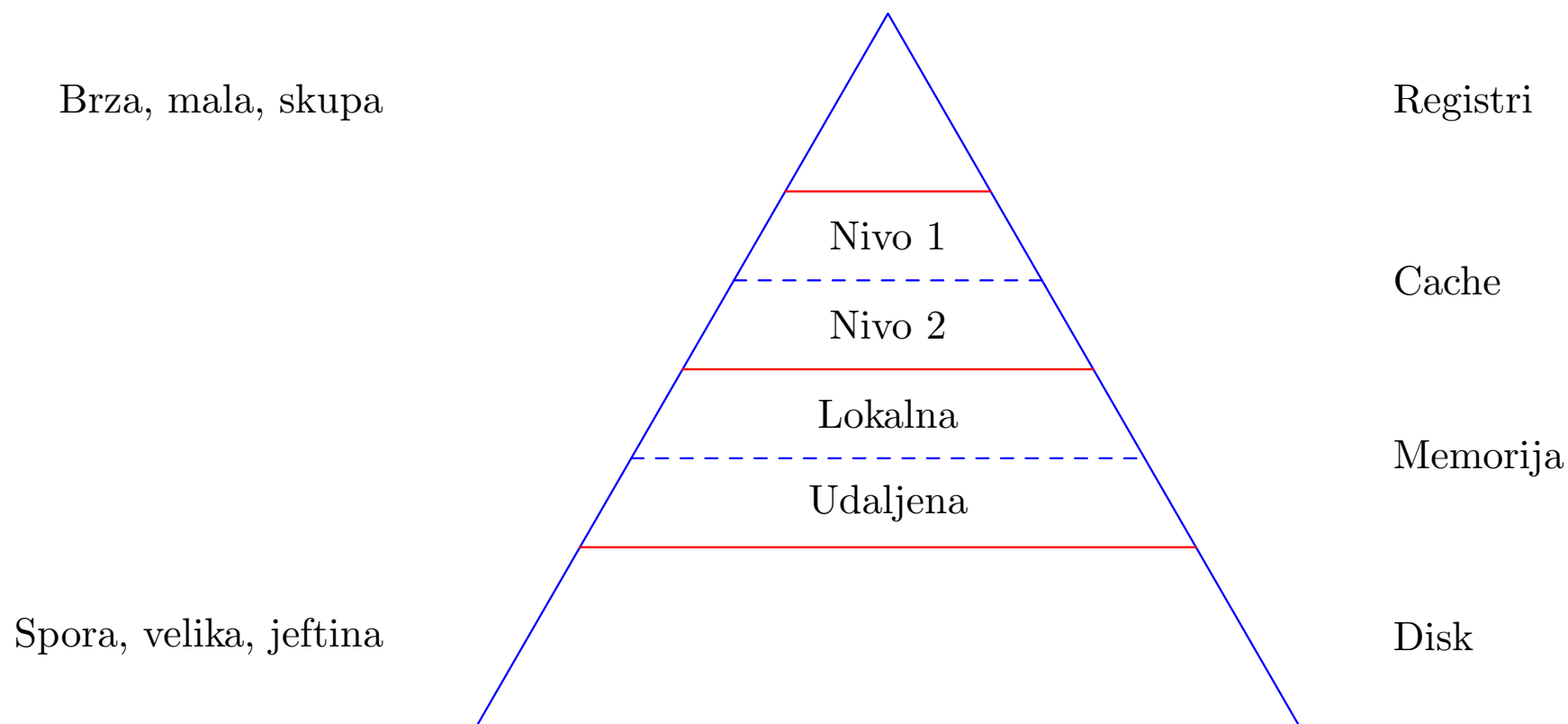
- Dodatnom hijerarhijskom strukturom memorije, između obične radne memorije (RAM) i registara procesora.

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**.