Programiranje 1
3. predavanje

Saša Singer

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

PMF – Matematički odjel, Zagreb

Sadržaj predavanja

- Građa računala:
  - Memorija (bistabil, bit, riječ, byte),
  - Procesor (registri, aritmetičko–logička jedinica, upravljačka jedinica),
  - Ulazna jedinica, izlazna jedinica.
- Stvarni "izgled" računala:
  - Registri modernog procesora (IA–32).
  - Primjer matične ploče, blok–dijagram.
  - Hijerarhijska struktura memorije (cache).
  - "Priča o cacheu".

Informacije

Ne zaboravite da treba:

- otvoriti korisnički račun u Računskom centru,
- utorkom i četvrtkom, od 11 do 13 sati.

Nadalje, treba:

- obaviti prijavu i dobiti potvrdu prijave

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

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

Prilikom prijave za "ku", svoje podatke trebate upisati korektno, što (između ostalog) znači i

- korištenje hrvatskih slova u imenu i prezimenu!

Informacije — nastavak

Studenti koji su upisali "czsdj" varijantu imena i prezimena neka se jave e-mailom asistentu V. Šegi na adresu

vsego@math.hr

i napišu

- svoj JMBAG i ispravno ime i prezime.

Na kolokvijima (posebno, prvom)

- dozvoljeno je imati kalkulator s osnovnim računskim operacijama,

ali ne i "pametniji" od toga!

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-tih) 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 $1 -$ 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, Core Duo, . . .).

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 500 MHz,

  ali, zasad, nisu stigle do 1 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.

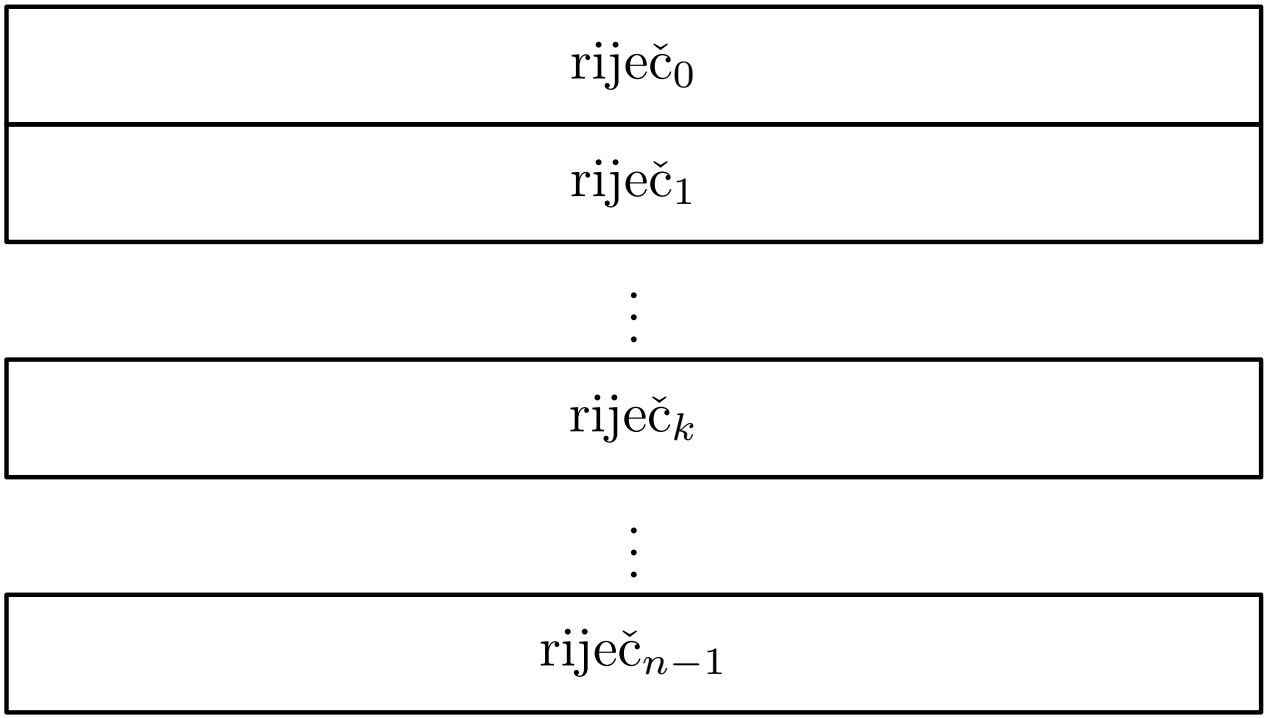
- Takve veće cjeline prikazuju potrebne vrste podataka i na njima kao cjelinama možemo izvršavati pripadne operacije i to brzo na nivou arhitekture računala.
- Koliko bitova čini jednu riječ? Ne postoji točan odgovor, jer to ovisi o tipu podataka koji se prikazuje u arhitekturi računala.
- Pojednostavljeno, 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)

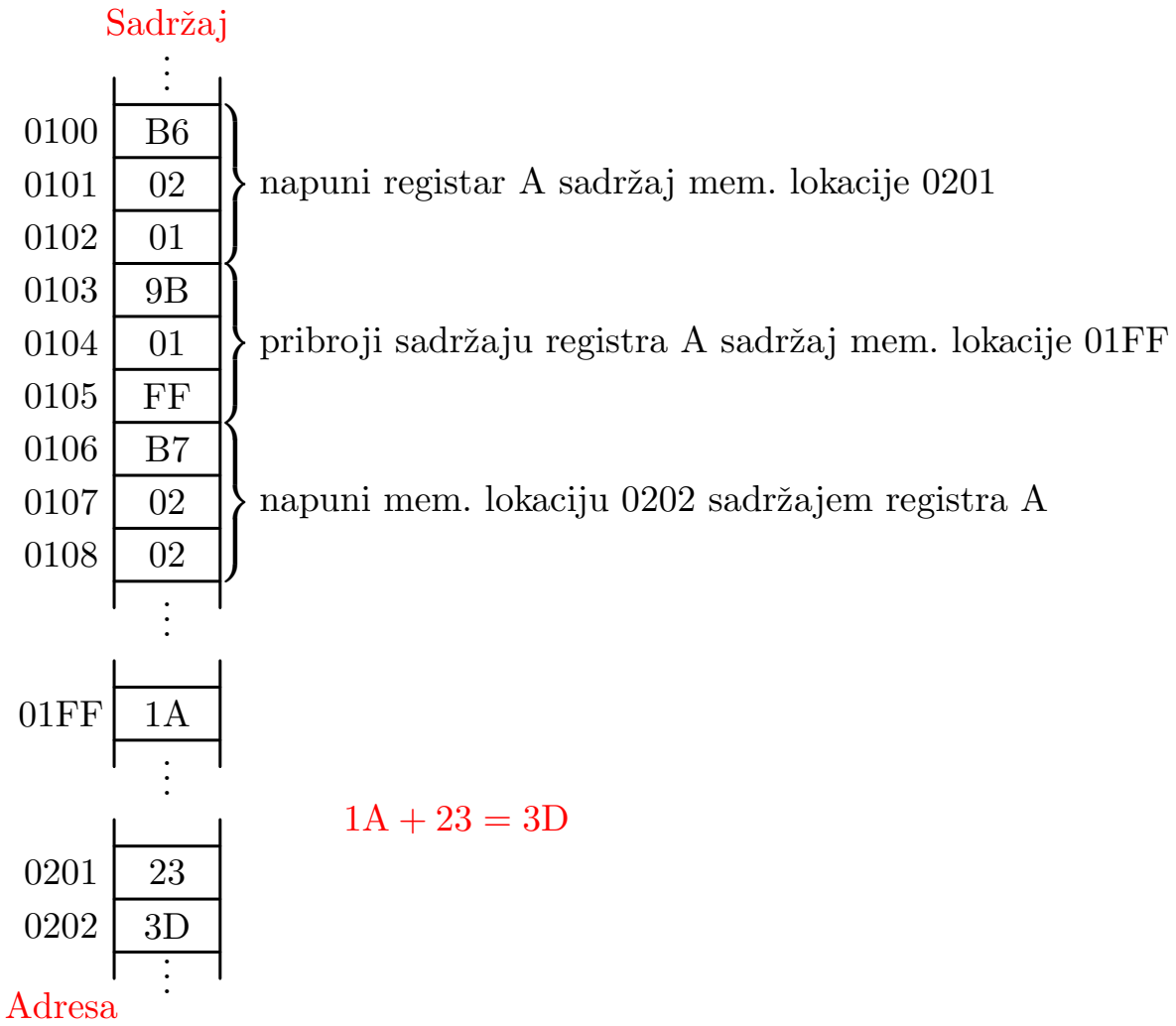
- Osim toga, procesor ima i skup registara. To je radna memorija procesora za spremanje:
  - podataka nad kojima se izvršavaju instrukcije i rezultata tih operacija,
  - instrukcija (odnosno, dijelova instrukcija) koje se izvršavaju.
- Sve operacije u procesoru mogu se napraviti
  - samo na operandima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

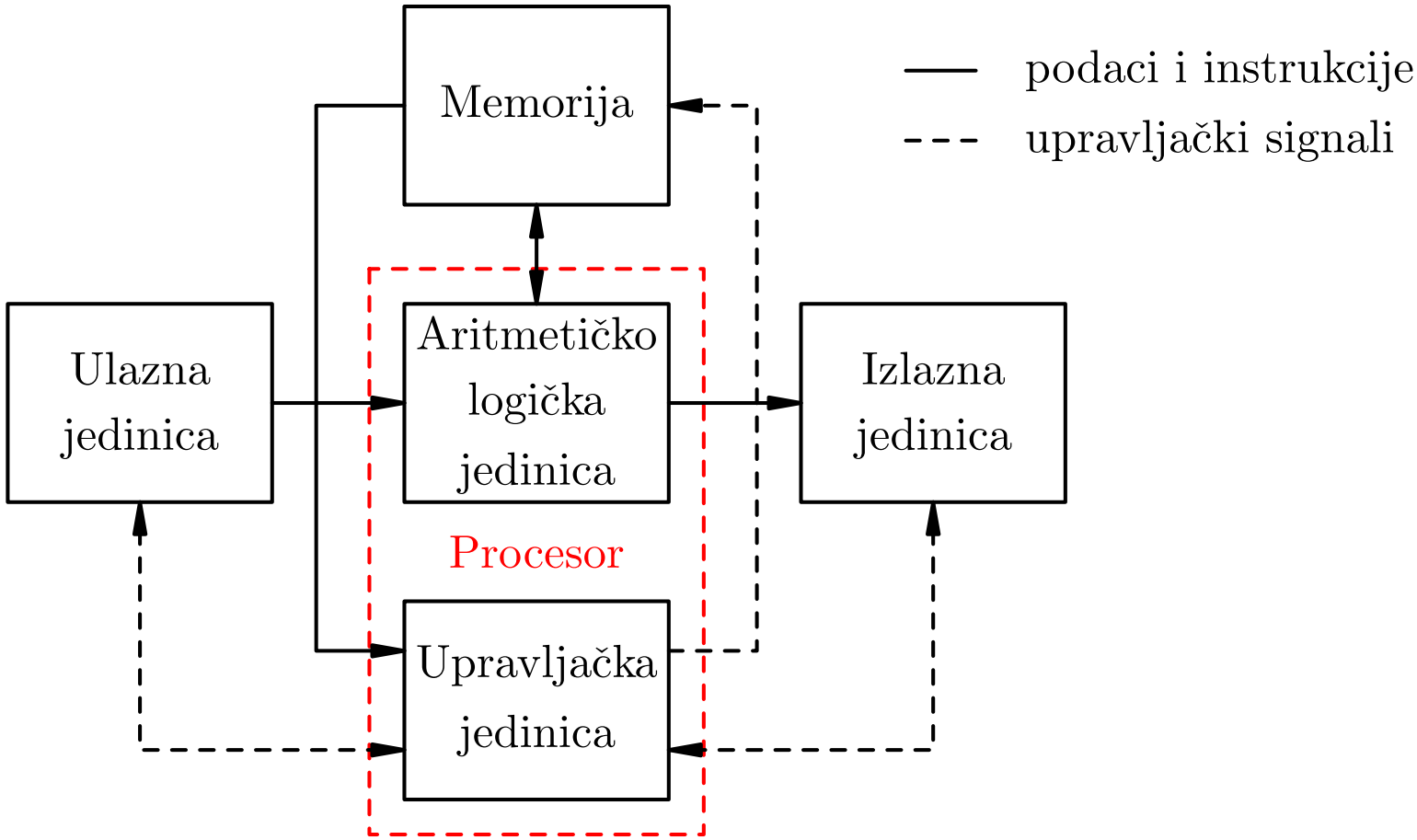


Ulazne i izlazne jedinice

- 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:



Stvarni "izgled" računala

Sadržaj

- Stvarni "izgled" računala:
  - Registri modernog procesora (IA–32).
  - Primjer matične ploče, blok–dijagram.
  - Hijerarhijska struktura memorije (cache).
  - "Priča o cacheu".

Standardni kućni procesori

Standardni kućni procesori bazirani su na tzv. IA–32 arhitekturi (Intel ili AMD, svejedno mi je). Osnovna svojstva:

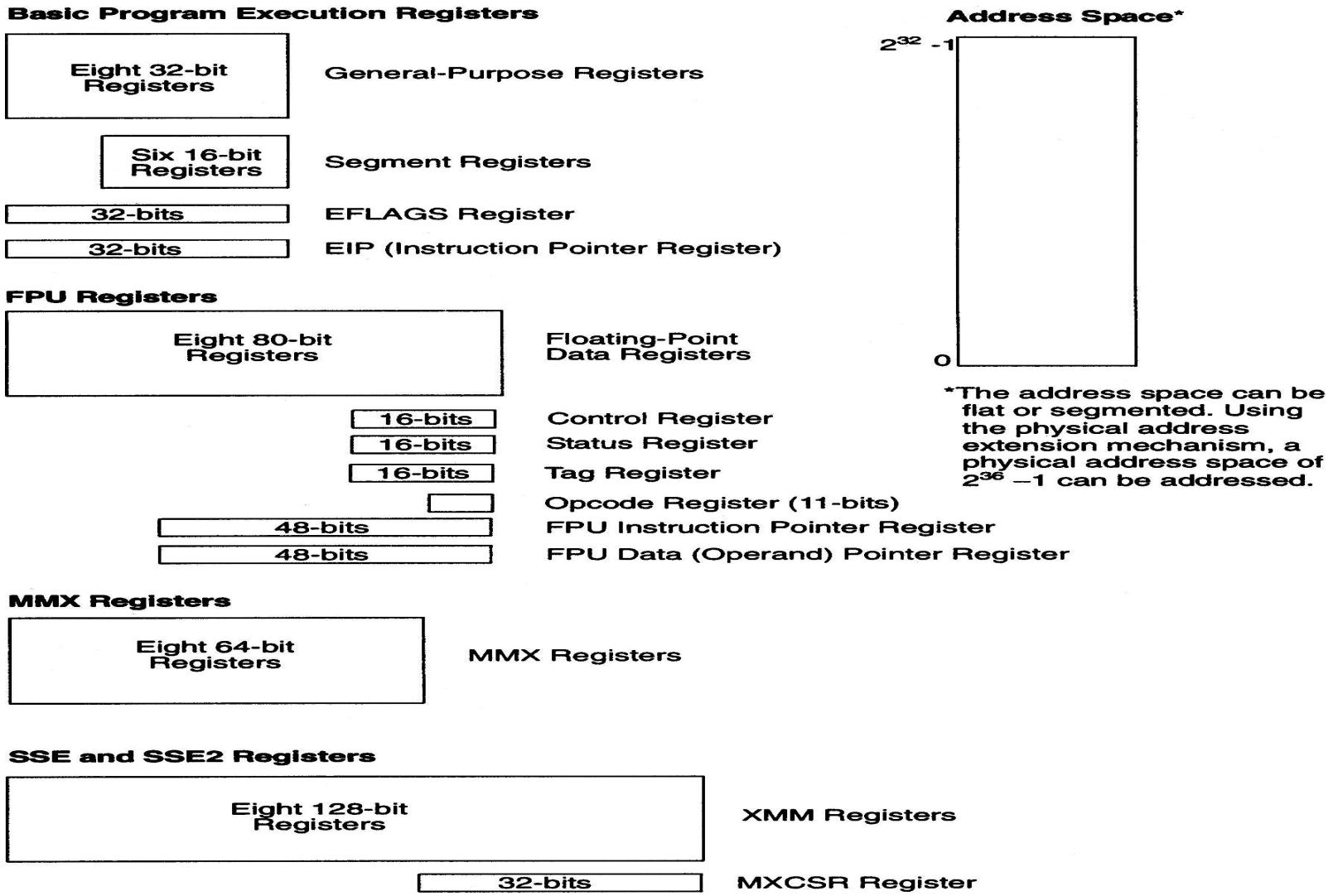
- riječ = 32 bita = 4 B, (toliki je tip `int` u C-u),
- adresa = 32 bita (x86) ili, modernije, 64 bita (x64).

Ovi procesori imaju gomilu registara, raznih namjena, koji sadrže razne vrste podataka i instrukcija (ili dijelova instrukcija).

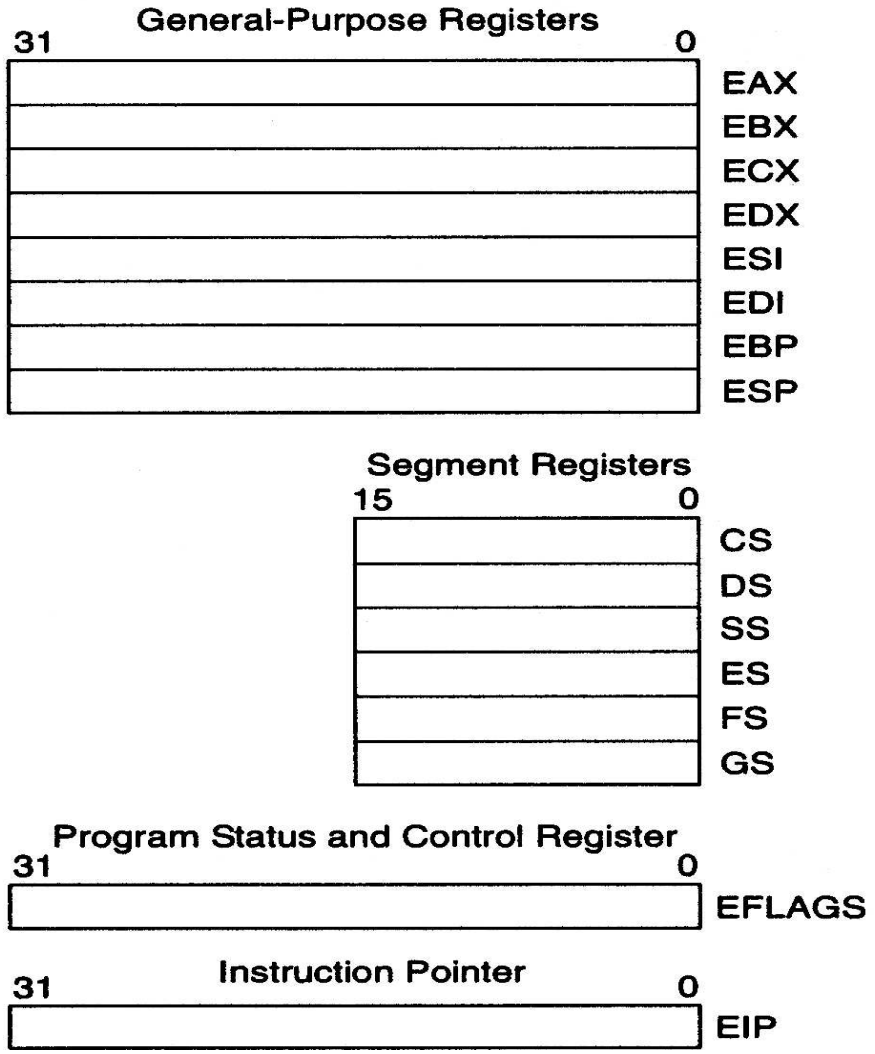
Shematski izgled svih registara, a onda samo registara opće namjene dan je na sljedeće dvije stranice.

Napomena: slike odgovaraju IA–32 procesoru Pentium 4, serija Northwood, podnožje 478 (danas već zastarjelom).

IA–32 — Svi registri i adresni prostor



IA–32 — Osnovni izvršni registri

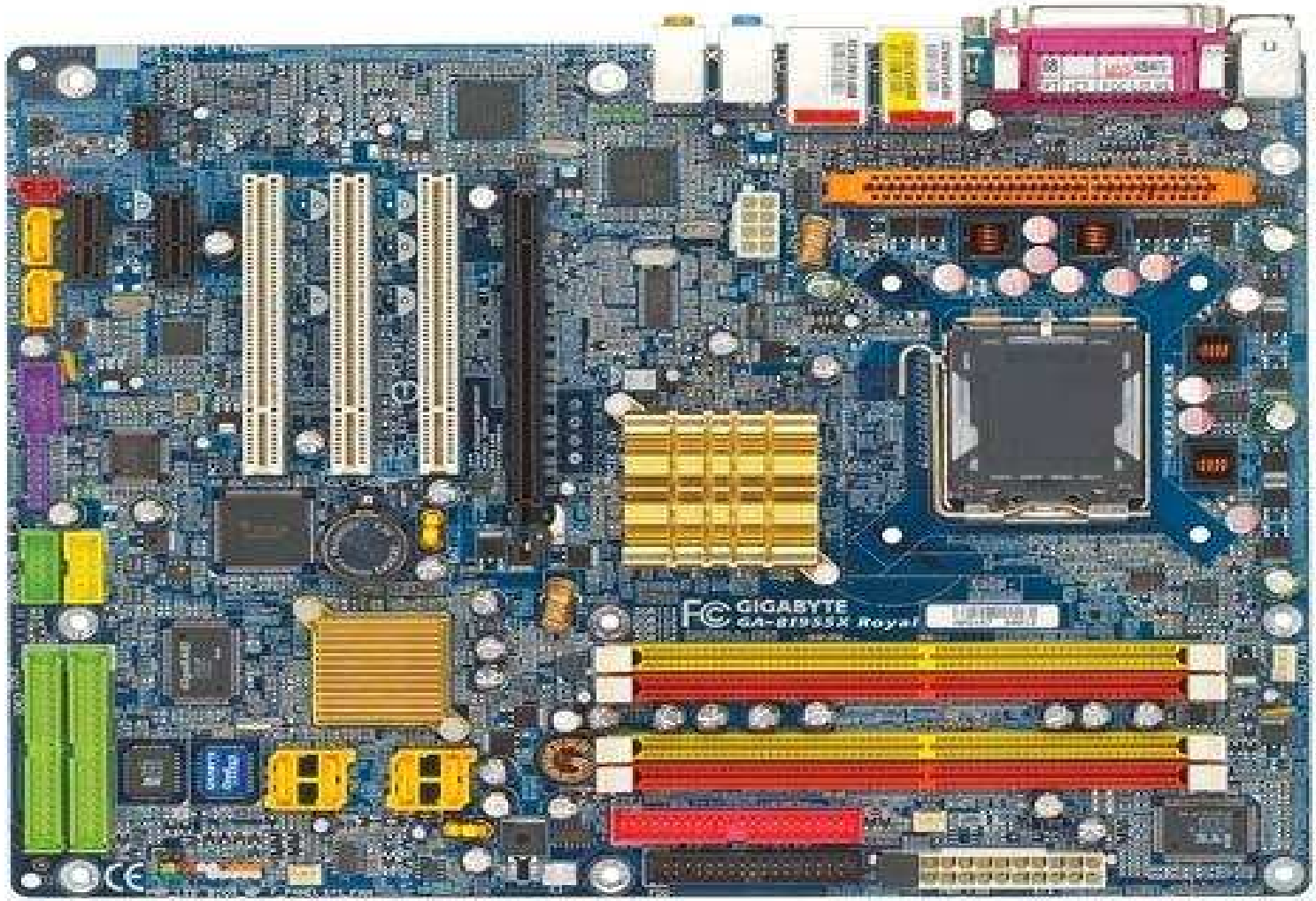


Izgled matične ploče računala

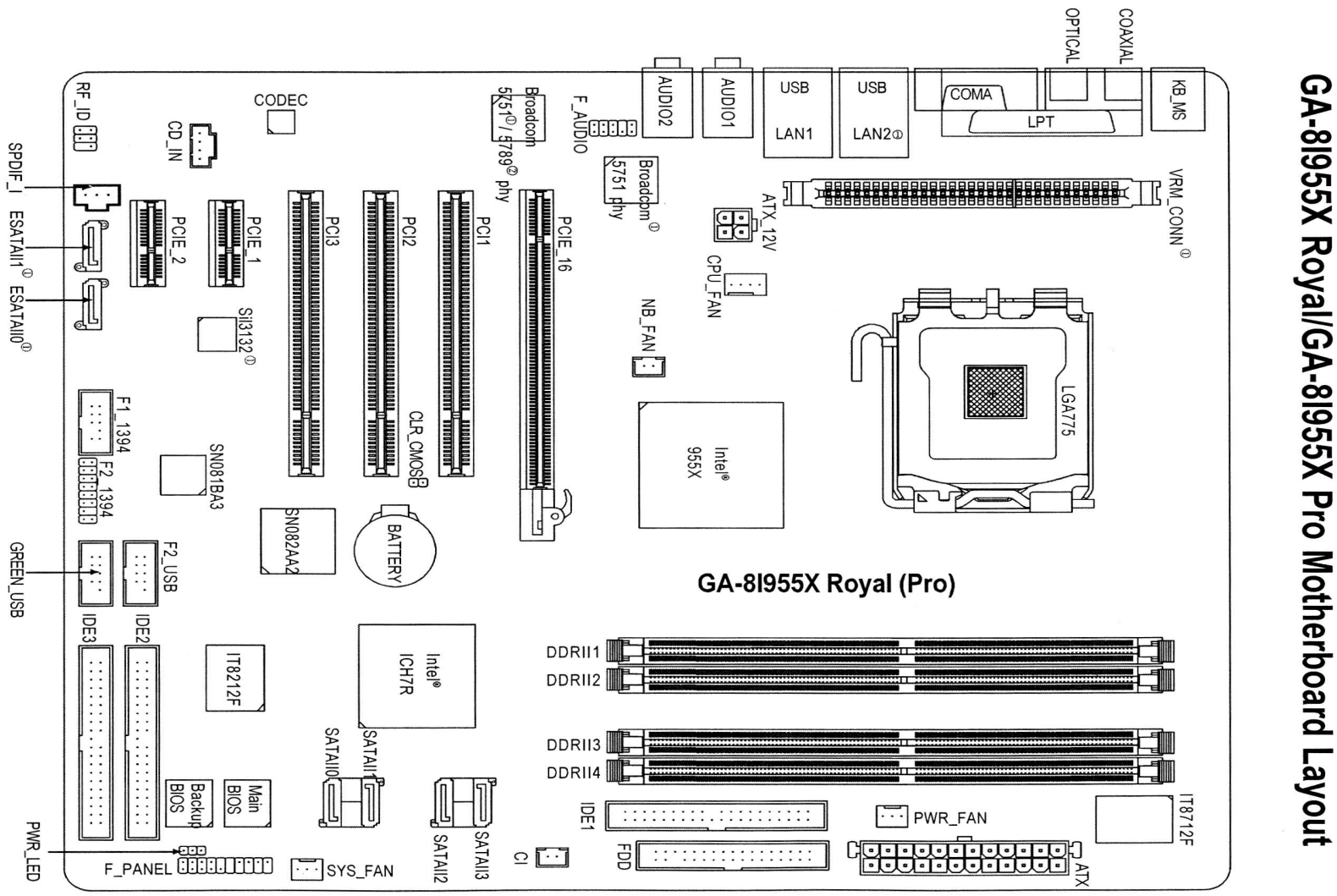
Moderna "kućna" računala, naravno, imaju sve standardne dijelove računala.

- 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.

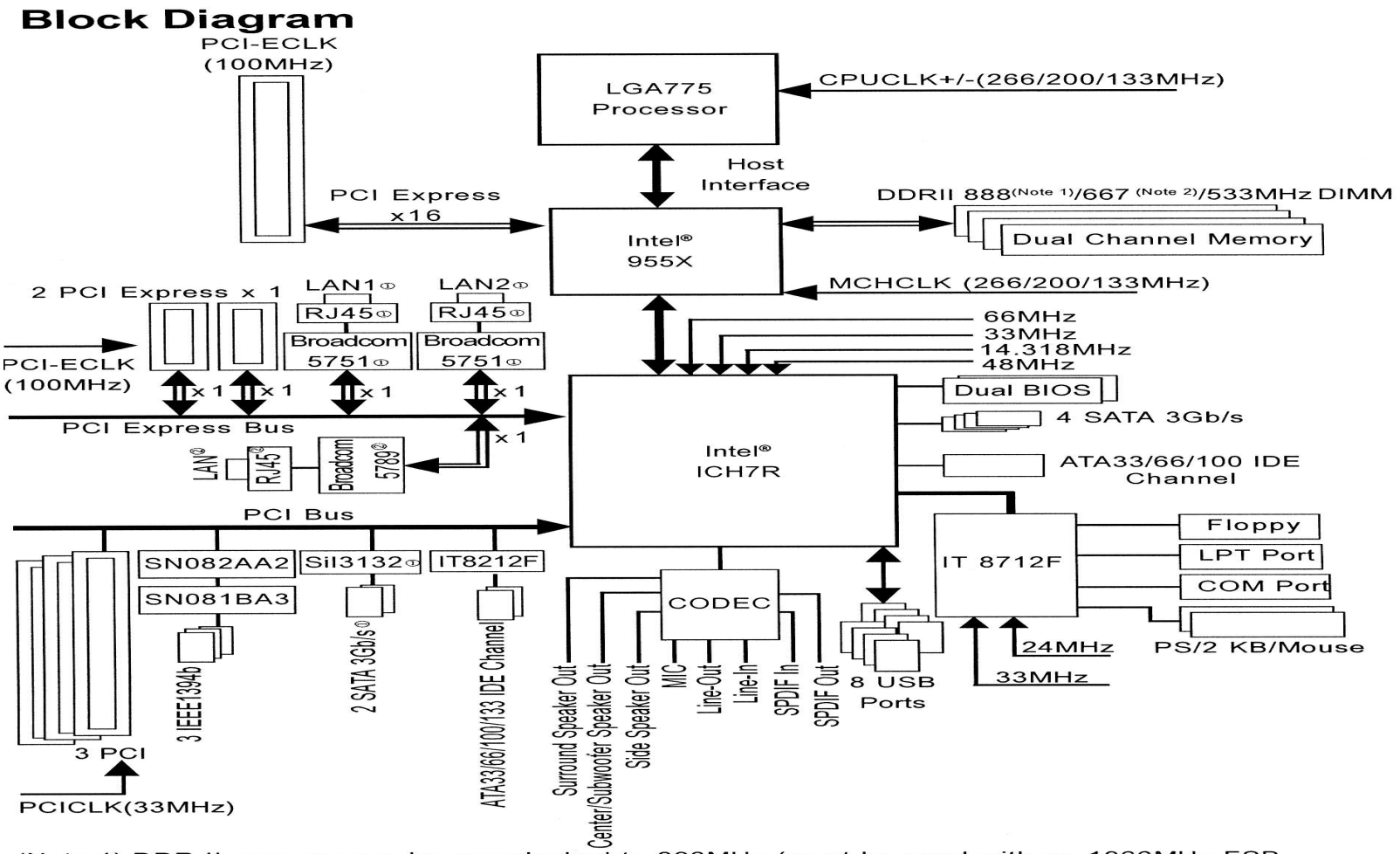
Matična ploča GA–8I955X Royal — izgled



Matična ploča — raspored



Matična ploča — blok dijagram



(Note 1) DDR II memory can be overclocked to 888MHz (must be used with an 1066MHz FSB processor) through overclocking in BIOS. Go to GIGABYTE's website for more information about the supported DDR II memory modules for this feature.
(Note 2) To use a DDR II 667 memory module on the motherboard, you must install an 800/1066MHz FSB processor.

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),
  - DVD i CD uređaji,
  - 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.

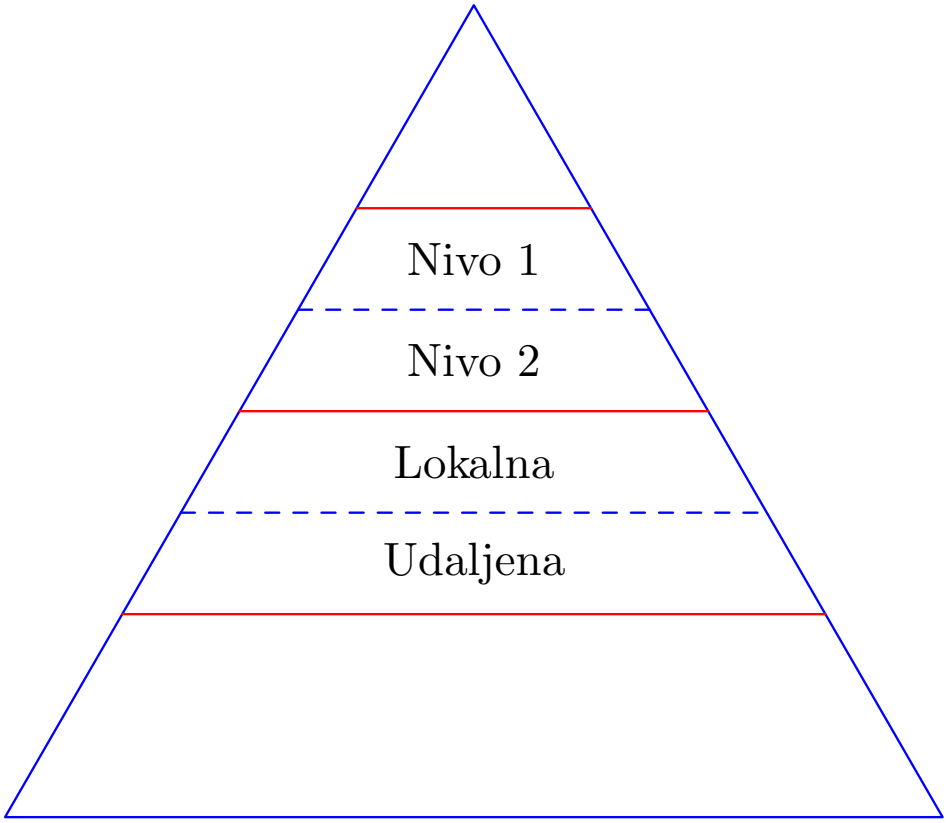
Kako se to izbjegava, ili, barem ublažava?

- 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.