Uvod u računarstvo
4. predavanje

Saša Singer

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

PMF – Matematički odjel, Zagreb

Sadržaj predavanja

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

Sadržaj predavanja (nastavak)

- Osnovni tipovi podataka u računalu (pregled):
  - višekratnici byte-a, odn. riječi (bez strukture),
  - cijeli brojevi bez predznaka,
  - cijeli brojevi s predznakom,
  - "realni" (floating–point) brojevi.
- Cijeli brojevi — prikaz i aritmetika:
  - adrese — "obična" aritmetika,
  - modularna aritmetika cijelih brojeva,
  - prikaz brojeva bez predznaka — sustav ostataka,
  - prikaz brojeva s predznakom — sustav ostataka,
  - tipične pogreške u korištenju cijelih brojeva.

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)

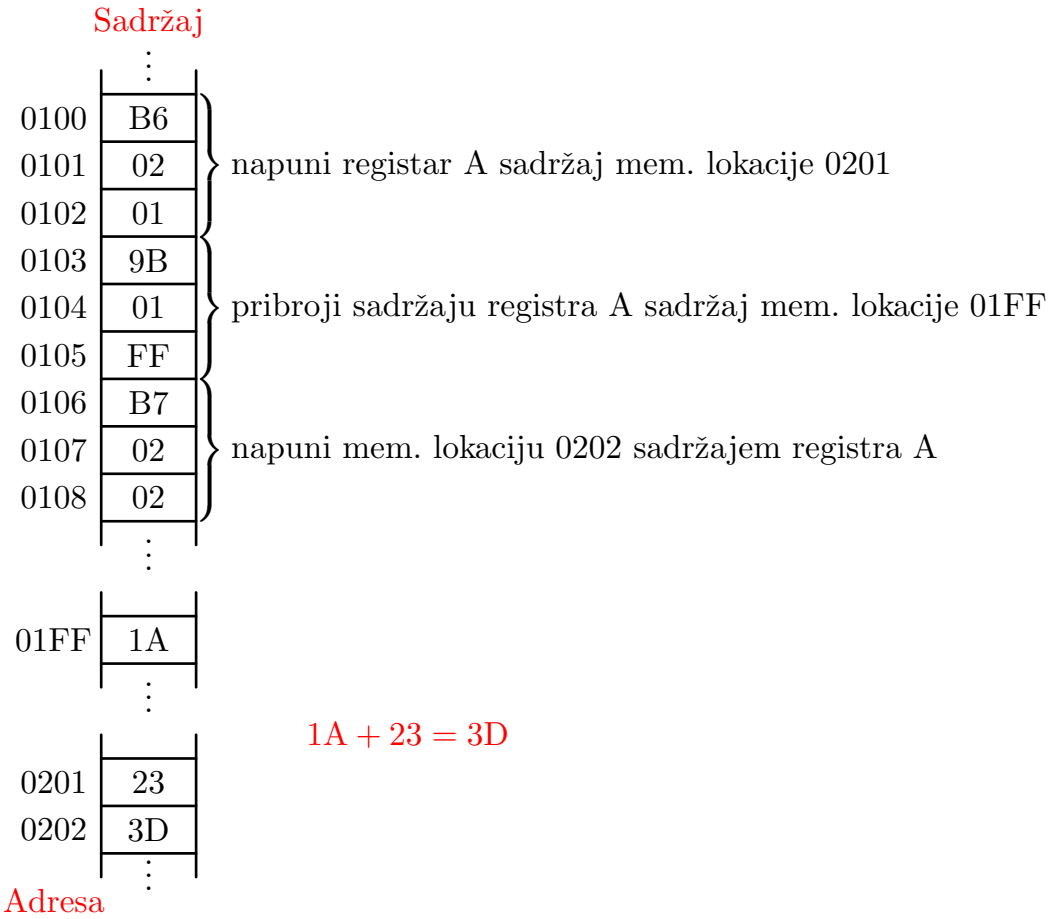
- Osim toga, procesor ima i skup registara. To je radna memorija procesora za spremanje:
  - podataka nad kojima se izvršavaju instrukcije i rezultata tih operacija,
  - instrukcija (odnosno, dijelova instrukcija) koje se izvršavaju.
- Sve operacije u procesoru mogu se napraviti
  - samo na operandima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

- Program:

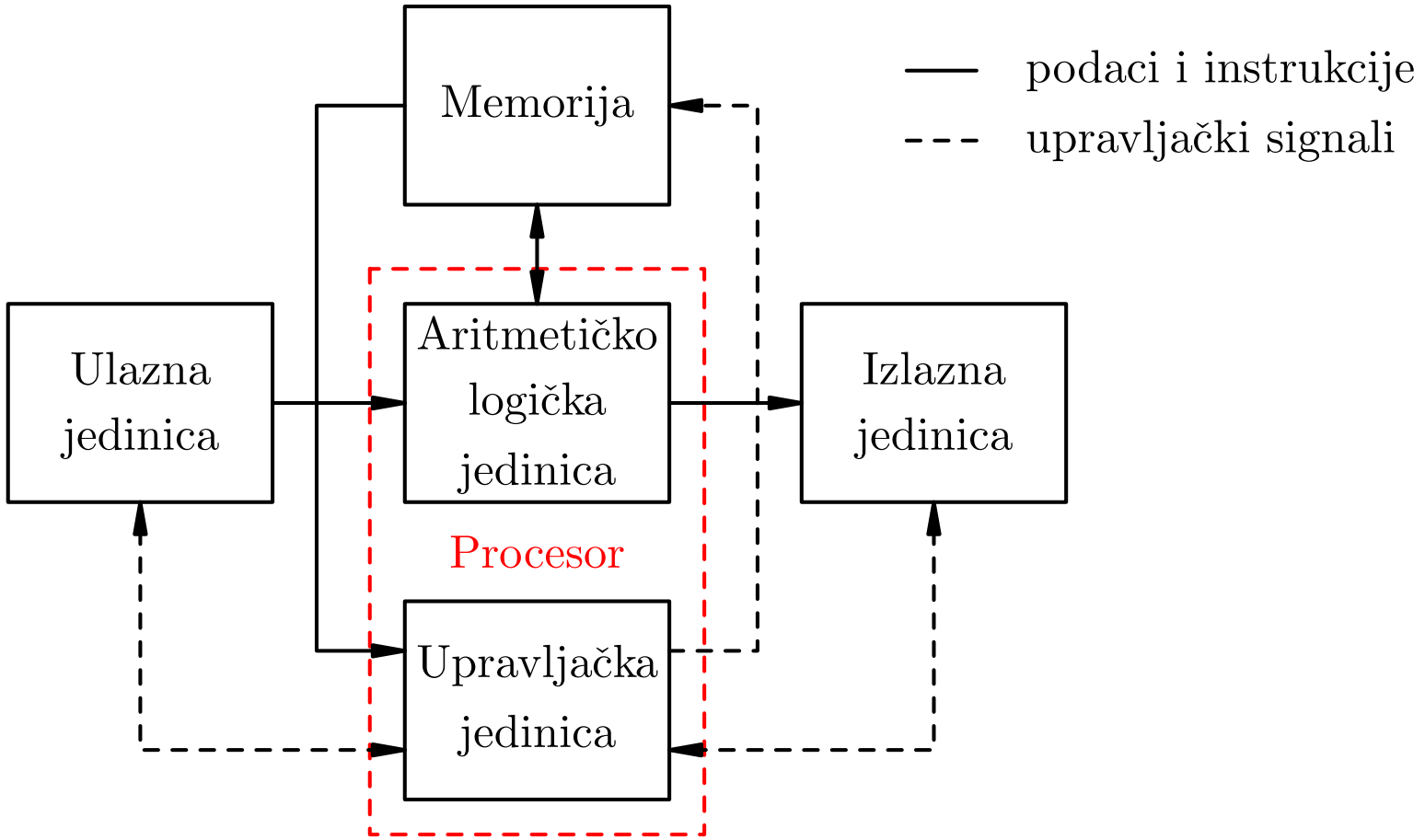


Ulazne i izlazne jedinice

- 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).

Standardni kućni procesori

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

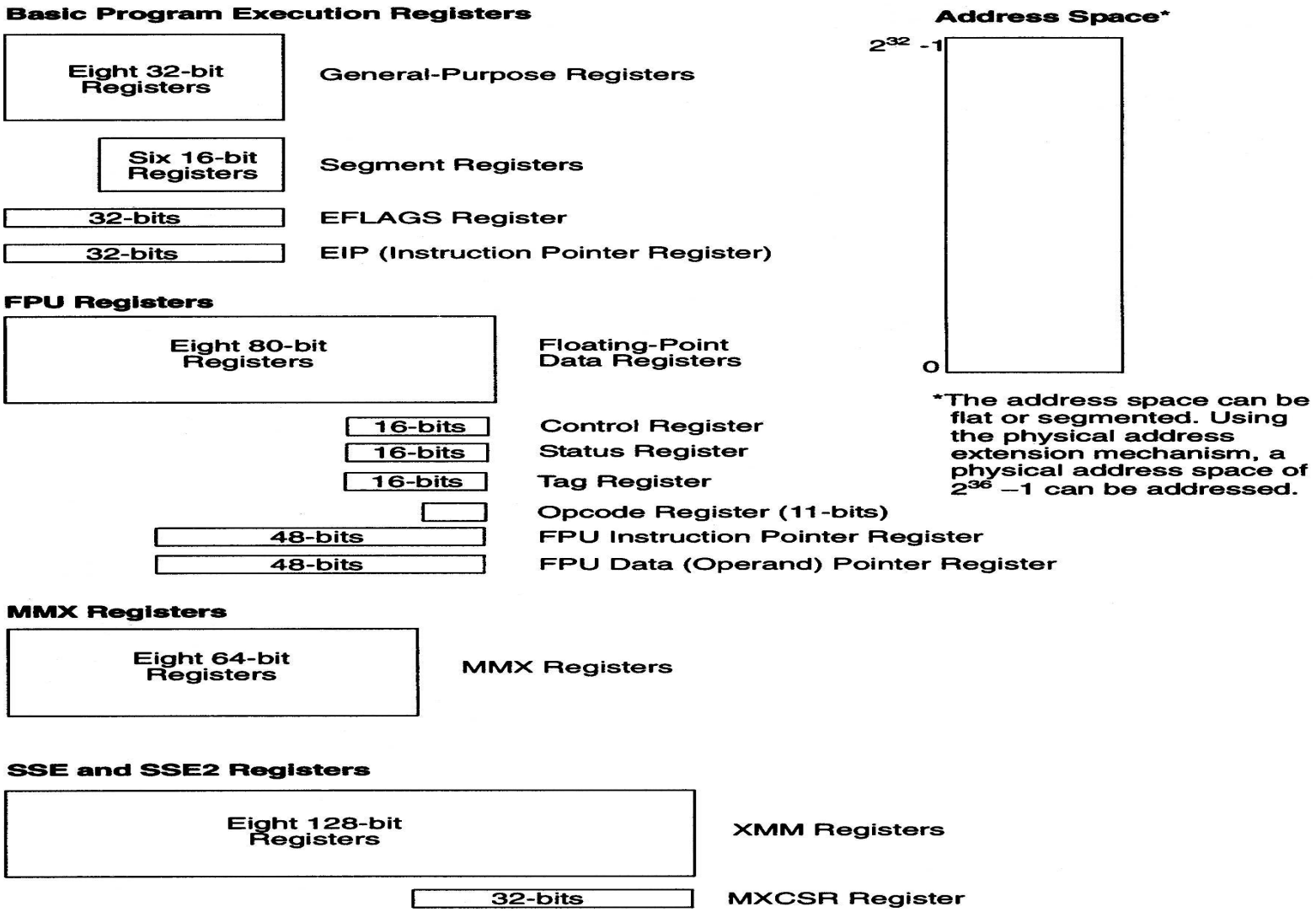
- riječ = 32 bita = 4 B, (toliki je tip `int` u C-u),
- adresa = 32 bita ili, modernije, 64 bita (x64).

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

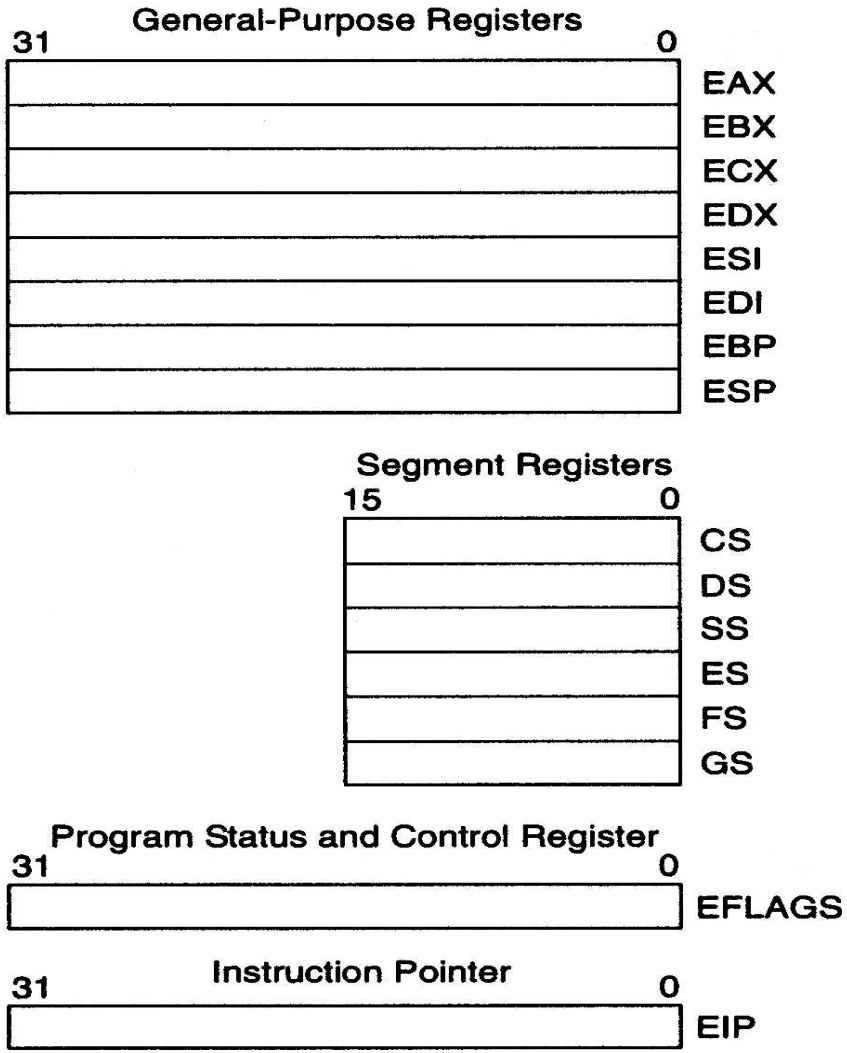
Shematski izgled svih registara, a onda samo registara opće namjene dan je na sljedeće dvije stranice.

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

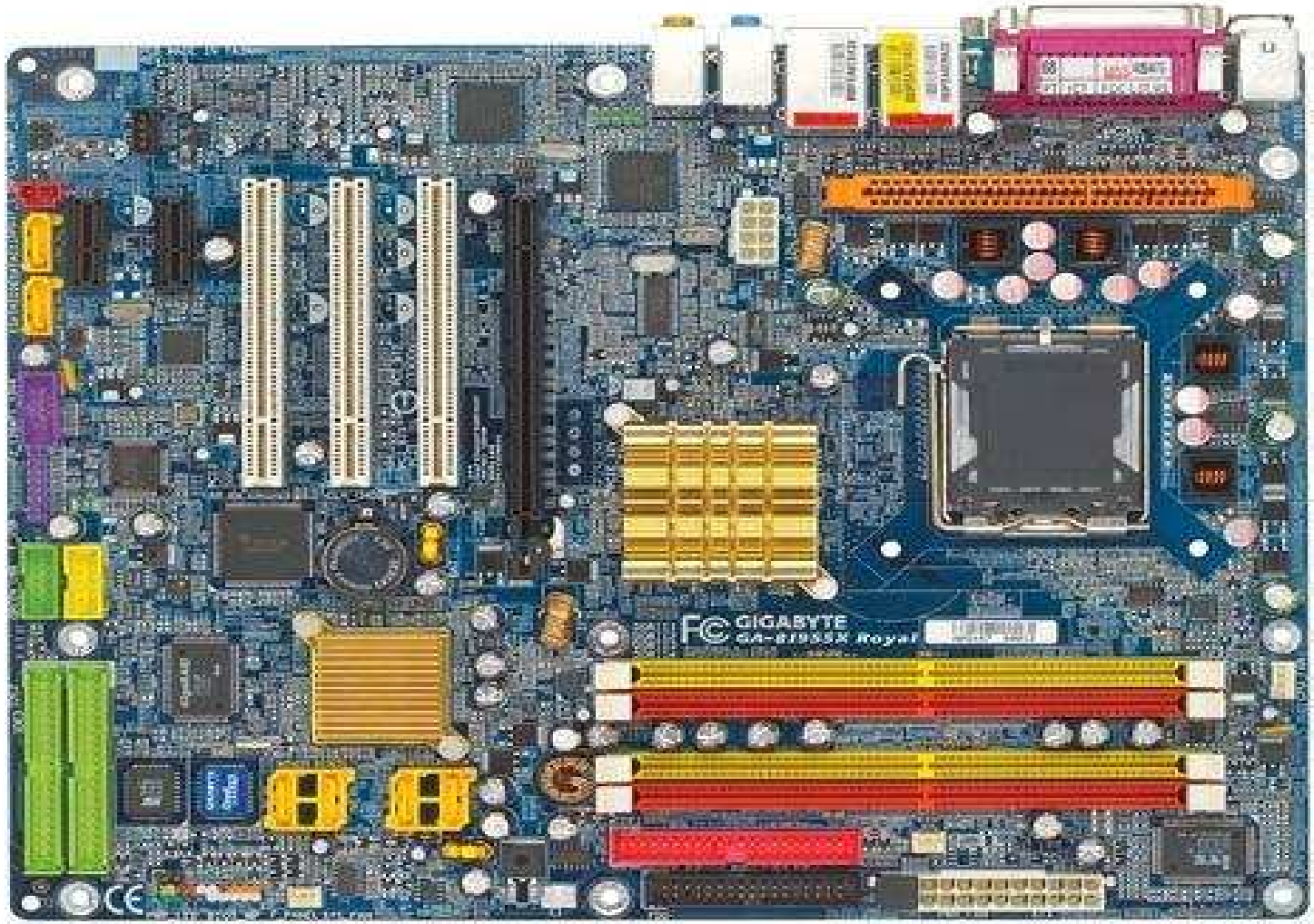


Izgled matične ploče računala

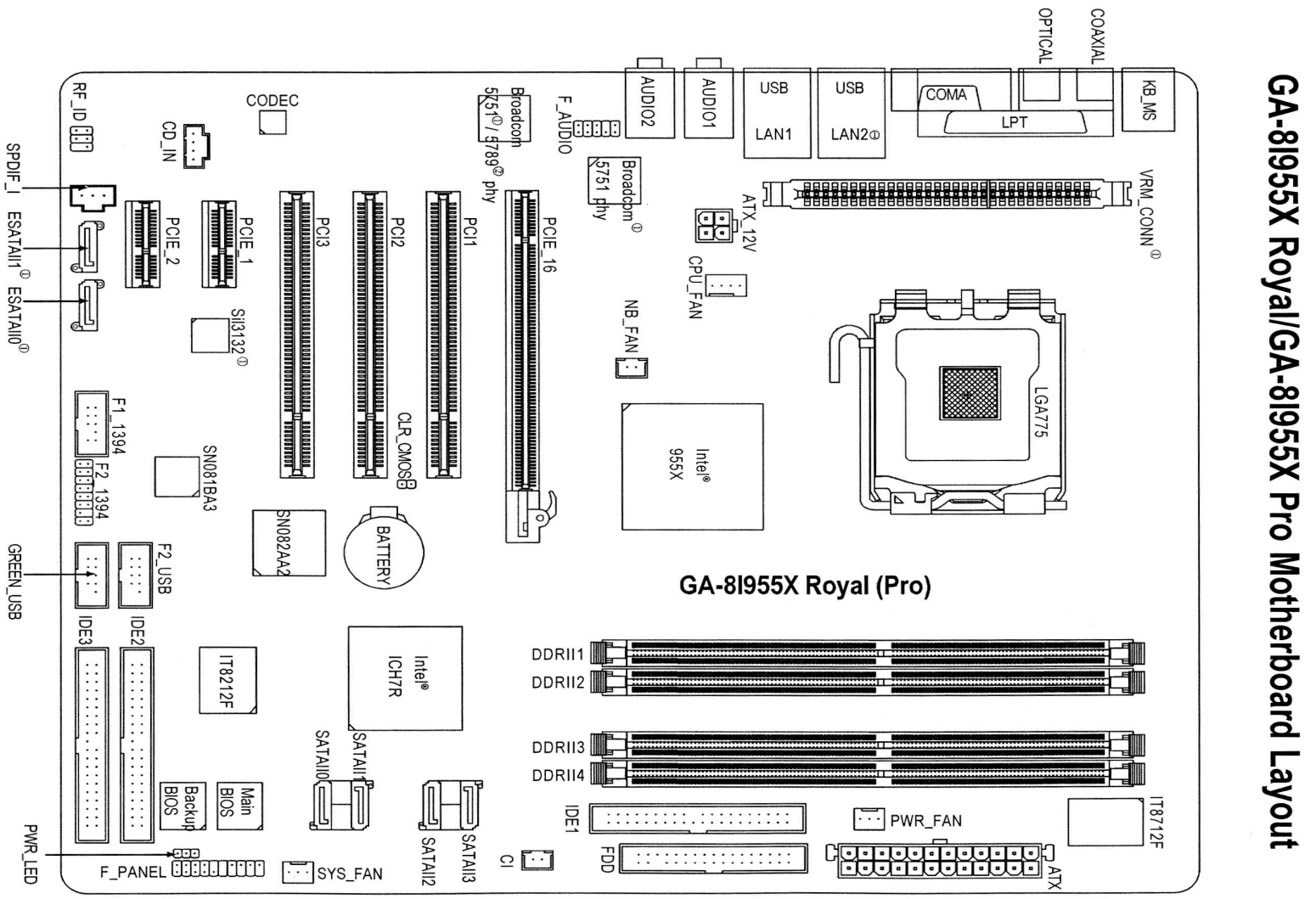
Moderna "kućna" računala, naravno, imaju sve standardne dijelove računala.

- 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.

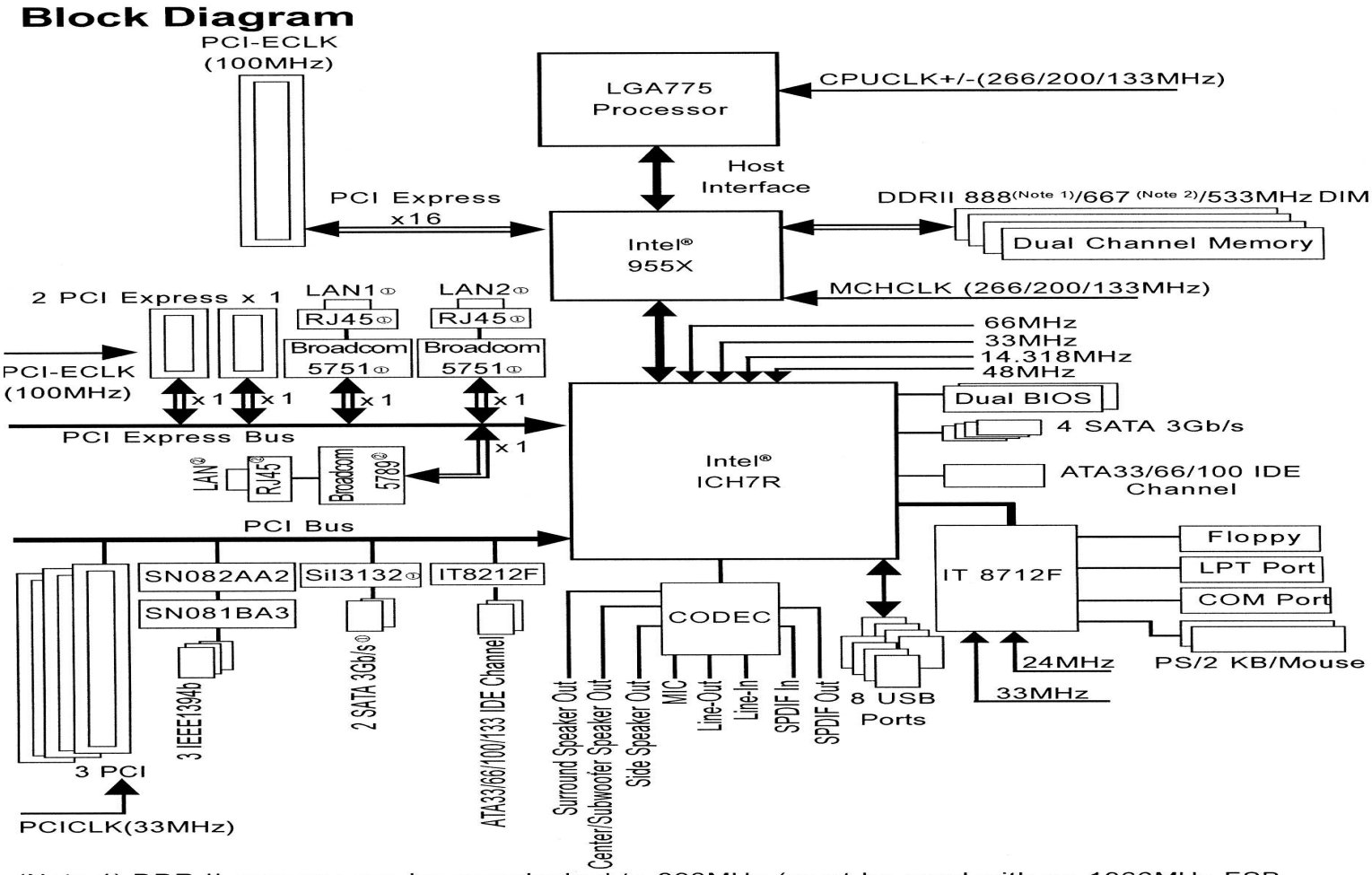
Matična ploča GA–8I955X Royal — izgled



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),
  - 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.

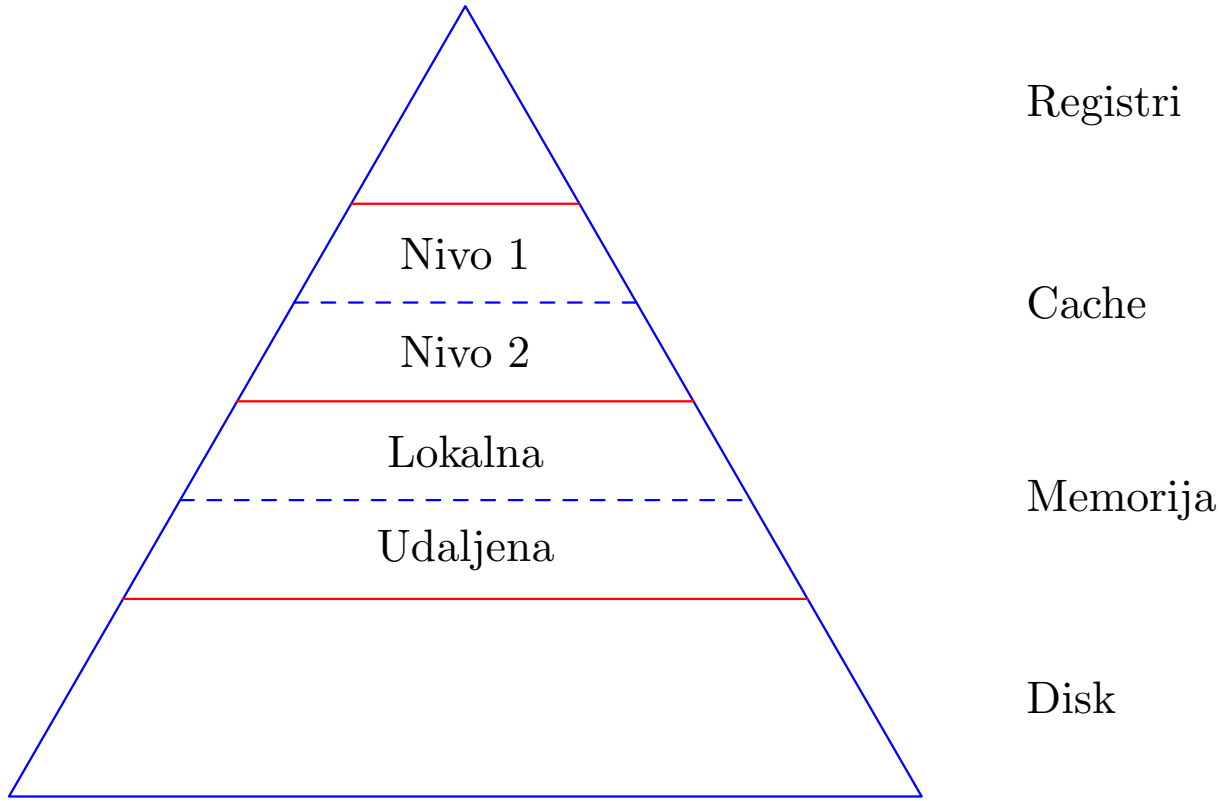
Kako se to izbjegava, ili, barem ublažava?

- 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.

Prikaz podataka u računalu

Sadržaj

- Osnovni tipovi podataka u računalu (pregled):
  - višekratnici byte-a, odn. riječi (bez strukture),
  - cijeli brojevi bez predznaka,
  - cijeli brojevi s predznakom,
  - "realni" (floating–point) brojevi.
- Cijeli brojevi — prikaz i aritmetika:
  - adrese — "obična" aritmetika,
  - modularna aritmetika cijelih brojeva,
  - prikaz brojeva bez predznaka — sustav ostataka,
  - prikaz brojeva s predznakom — sustav ostataka,
  - tipične pogreške u korištenju cijelih brojeva.

Osnovni tipovi podataka u računalu

Jednostavno rečeno, osnovni ili fundamentalni tipovi podataka u računalu su:

- one cjeline ili blokovi bitova s kojima računalo "zna nešto raditi", i to neovisno o njihovom sadržaju.

To znači da postoje instrukcije koje nešto rade s tim cjelinama kao operandima, bez obzira na eventualni dodatni tip operanda. U ovom kontekstu je

- tip = interpretacija sadržaja.

Tipični primjer takvih instrukcija su

- instrukcije za transfer tih cjelina između memorije i registara procesora.

Osnovni tipovi podataka (nastavak)

Ako se sjetimo "pravokutnog" izgleda memorije, onda u te tipove sigurno ulaze

- osnovne cjeline koje možemo adresirati,

dakle, ono što smo ranije (u skici memorije) nazvali riječ.

Napomena: kasnije smo pojam "riječ" koristili za nešto drugo — prostor za cijele brojeve, odnosno, instrukcije.

Osim toga, ovisno o veličini "osnovne stranice" (jedne adrese) memorije, računalo može "znati" raditi i s

- manjim cjelinama — dijelovima osnovne cjeline, ako je ona dovoljno velika,
- većim cjelinama — blokovima osnovnih cjelina.

Osnovni tipovi podataka (nastavak)

Vidimo da ti osnovni tipovi vrlo ovise o arhitekturi računala.

Obzirom na to da sadržaj nije bitan, zapravo jedino što se može reći o tim cjelinama je

- njihova duljina — u bitovima.

Naravno, imena ili nazivi za cjeline istih duljina variraju na raznim arhitekturama.

Za nas kao korisnike, ovi tipovi nisu naročito važni, ali

- zgodno ih je upoznati (navesti njihove duljine i nazive), barem na jednom primjeru.

Primjer: Intelova 32-bitna arhitektura procesora (IA–32).

Osnovni tipovi podataka na IA–32

Osnovna cjelina koju možemo adresirati na IA–32 je

1 byte = 8 bitova.

To je duljina "osnovne stranice" (jedne adrese) memorije.

Ostali osnovni tipovi podataka na IA–32 su većih duljina:

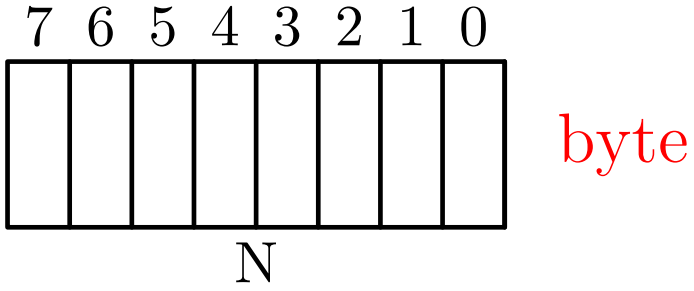
Naziv	Duljina (bitova)	Broj byteova
word	16	2
doubleword	32	4
quadword	64	8
double quadword	128	16

Označavanje bitova i adresa

Na primjeru osnovnih tipova podataka zgodno je odmah uvesti još dvije stvari:

- standardne oznake za bitove unutar pojedine cjeline (koriste se općenito, a ne samo na IA–32),
- način adresiranja pojedinih dijelova cjeline, obzirom na raspored bitova (specifično za pojedinu arhitekturu).

Jedan byte, duljine $n = 8$ bitova, na adresi N označavamo ovako (svaka "kućica" je 1 bit):



Označavanje bitova i adresa (nastavak)

Objašnjenje oznaka: Uzmimo da neka cjelina (u ovom slučaju, osnovni tip podataka) ima duljinu od n bitova.

- Tradicionalno se pojedini bitovi u cjelini indeksiraju slično kao i riječi u memoriji, dakle, od 0 do $n-1$.

Međutim, ovdje poredak ide "naopako", tako da

- "najdesniji" bit ima indeks 0 — najniži ili zadnji bit,
- "najljeviji" bit ima indeks $n-1$ — najviši ili vodeći bit.

Razlog: pozicioni prikaz cijelih brojeva (u bazi 2), u kojem vodeću znamenku (bit) pišemo kao prvu (lijevu), a najnižu znamenku (bit) pišemo kao zadnju (desnu). Osim toga, indeksi bitova odgovaraju pripadnim potencijama baze 2.

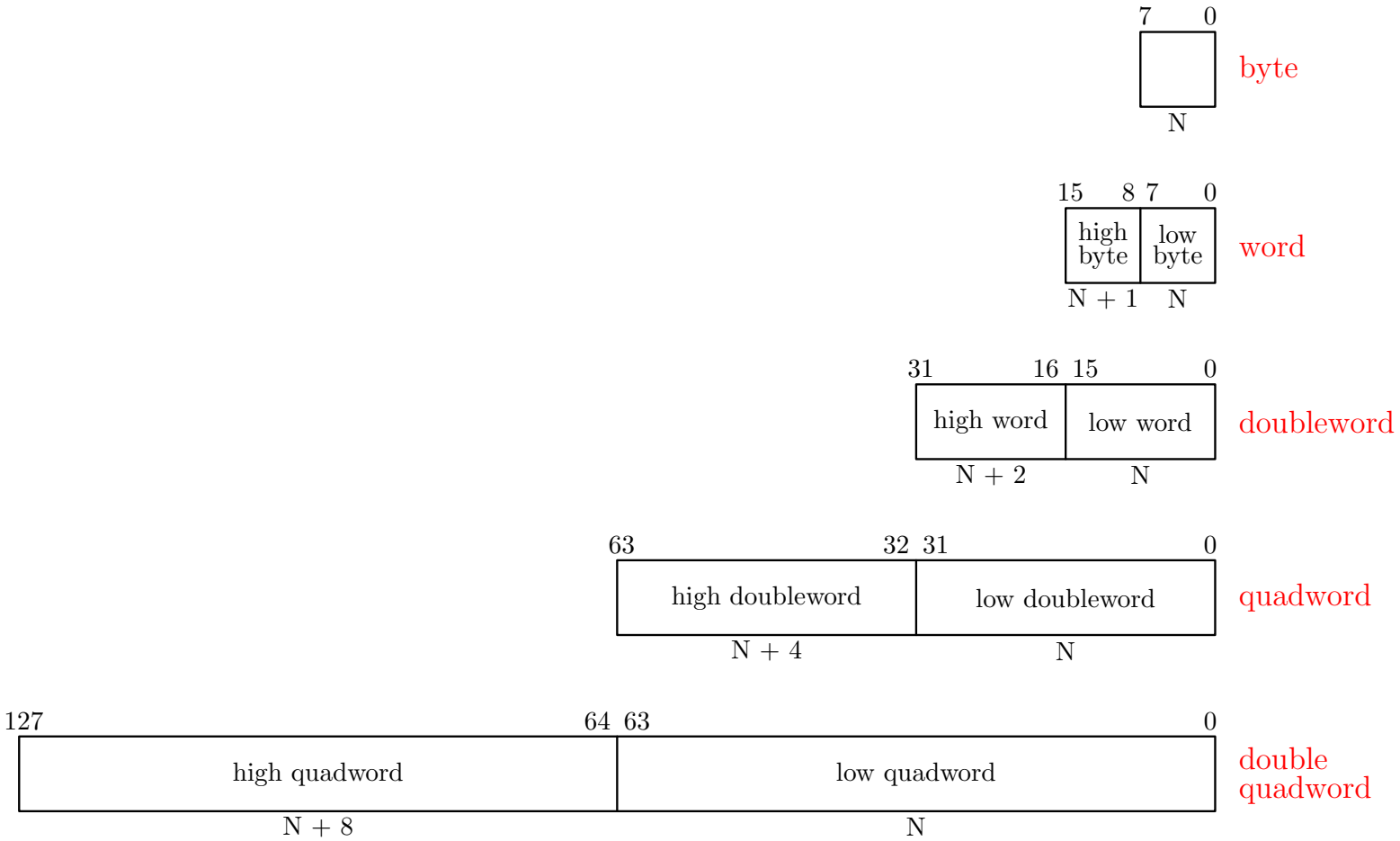
Detalji malo kasnije!

Označavanje bitova i adresa (nastavak)

- Ove oznake za bitove katkad pišemo iznad, a katkad ispod skice cjeline (ali značenje je uvijek očito iz slike).
- Na početku ih pišemo iznad, jer ispod skice pišemo adrese na kojima se nalaze pojedini adresabilni dijelovi cjeline — u ovom slučaju byteovi (adresabilne cjeline na IA–32 su byteovi).
- Na IA–32, najniži byteovi su i na najnižim adresama, ali postoje arhitekture računala na kojima je obratno (vodeći dio je na najnižoj adresi).
- Dogovorno, najniža adresa (na kojoj "počinje" cjelina) je ujedno i adresa čitave cjeline (bez obzira na poredak dijelova).

Osnovni tipovi podataka na IA–32 (nastavak)

Osnovni tipovi podataka na IA–32 onda izgledaju ovako:



Jednostavni tipovi podataka

Ponovimo, ovi osnovni tipovi podataka nisu jako korisni, jer s njima ne možemo ništa "pametnije" raditi, osim transfera.

Za stvarno "računanje", trebamo

- dodatnu interpretaciju sadržaja (bitova) cjeline i
- operacije s takvom vrstom podataka (cjelinom bitova).

Skup podataka i operacije na njima čine neku algebarsku strukturu koju zajedničkim imenom zovemo tip podataka.

Oni tipovi podataka za koje računalo "zna" ili može:

- prikazati pripadni skup podataka i
- izvesti pripadne operacije na njima,

zovu se jednostavni tipovi podataka.

Jednostavni tipovi podataka (nastavak)

Pojam "jednostavni" znači da su operacije na toj vrsti podataka izravno podržane arhitekturom računala, tj.

- postoje instrukcije za njih.

Dakle, te operacije su elementarne operacije (za računalo kao izvršitelja) i u principu su brze.

Standardne jednostavne tipove podataka možemo grubo podijeliti u dvije grupe:

- nenumerički tipovi — znakovi, logička algebra,
- numerički tipovi — "cijeli" i "realni" brojevi (nekoliko raznih tipova za obje vrste brojeva).

Vidjet ćemo da se nenumerički tipovi zapravo svode na numeričke.

Nenumerički tipovi podataka (pregled)

Ukratko o nenumeričkim tipovima podataka.

Znakovi:

- prikaz je u nekom kôdu (obično nadskup ASCII kôda), tj. cijelim brojevima,
- posebnih operacija sa znakovima (osim transfera) nema. Sve funkcije na znakovima svode se na elementarne operacije na cijelim brojevima (vidi C).

Dakle, znakovi nisu izravno vezani za arhitekturu računala, ali su jednostavni tip u operacijskim sustavima i programskim alatima.

Uglavnom služe za ulaz i izlaz, te kao dijelovi složenijih struktura podataka.

Znakovi (primjer)

Znakovni tip u programskom jeziku C zove se `char`.

```
#include <stdio.h>

int main(void) {
    char c='1';

    printf("%c\n", c);  /* 1 */
    printf("%d\n", c);  /* 49 */

    return 0;
}
```

Nenumerički tipovi podataka (pregled)

Logička ili Booleova algebra:

- logičke vrijednosti prikazuju se bitovima,

$$\text{laž} = 0, \quad \text{istina} = 1,$$

  koje opet možemo uzeti i kao cijele brojeve,
- osnovne operacije ne, i, ili (engl. not, and, or), svode se na aritmetičke u bazi 2.

Logičke operacije mogu se izvesti i bit–po–bit na čitavim skupinama bitova u nekoj većoj cjelini (vidi C).

Za nas kao korisnike, logička algebra služi za formulaciju i kombiniranje uvjeta u uvjetnim naredbama.

Logičke vrijednosti (primjer)

C nema posebni tip za logičke vrijednosti.

```
#include <stdio.h>

int main(void) {
    int i=10, j=20;

    printf("%d\n", i<j);   /* 1 */
    printf("%d\n", i>=j);  /* 0 */
    printf("%d\n", i==j);  /* 0 */

    return 0;
}
```

Numerički tipovi podataka (pregled)

Numerički tipovi podataka moraju realizirati

- četiri osnovne aritmetičke operacije na raznim skupovima brojeva.

Osnovni problem: standardni skupovi brojeva u matematici

$$\mathbb{N}, \quad \mathbb{Z}, \quad \mathbb{Q}, \quad \mathbb{R}$$

su beskonačni i ne možemo ih prikazati u računalu.

Umjesto toga, u računalu možemo prikazati samo neke konačne podskupove odgovarajućeg matematičkog skupa. Drugim riječima,

- konačni podskup je "model" beskonačnog skupa.

Numerički tipovi podataka (pregled)

Numeričke tipove možemo podijeliti u tri grupe, prema beskonačnom skupu kojeg "modeliramo":

- "cijeli" brojevi bez predznaka — model za $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$,
- "cijeli" brojevi s predznakom — model za $\mathbb{Z}$,
- "realni" brojevi — model za $\mathbb{R}$.

Navodnici naglašavaju da su pripadni "prikazivi" skupovi brojeva konačni.

Dodatno, svaka grupa ima nekoliko "podtipova", ovisno o "veličini" pripadnog konačnog skupa prikazivih brojeva.

Numerički tipovi podataka (pregled)

Osim toga, prijelaz na konačne skupove bitno mijenja realizaciju aritmetike na odgovarajućem skupu. Aritmetika se ne nasljeđuje projekcijom s originalnog skupa!

Za potpuni opis numeričkih tipova podataka, moramo još opisati:

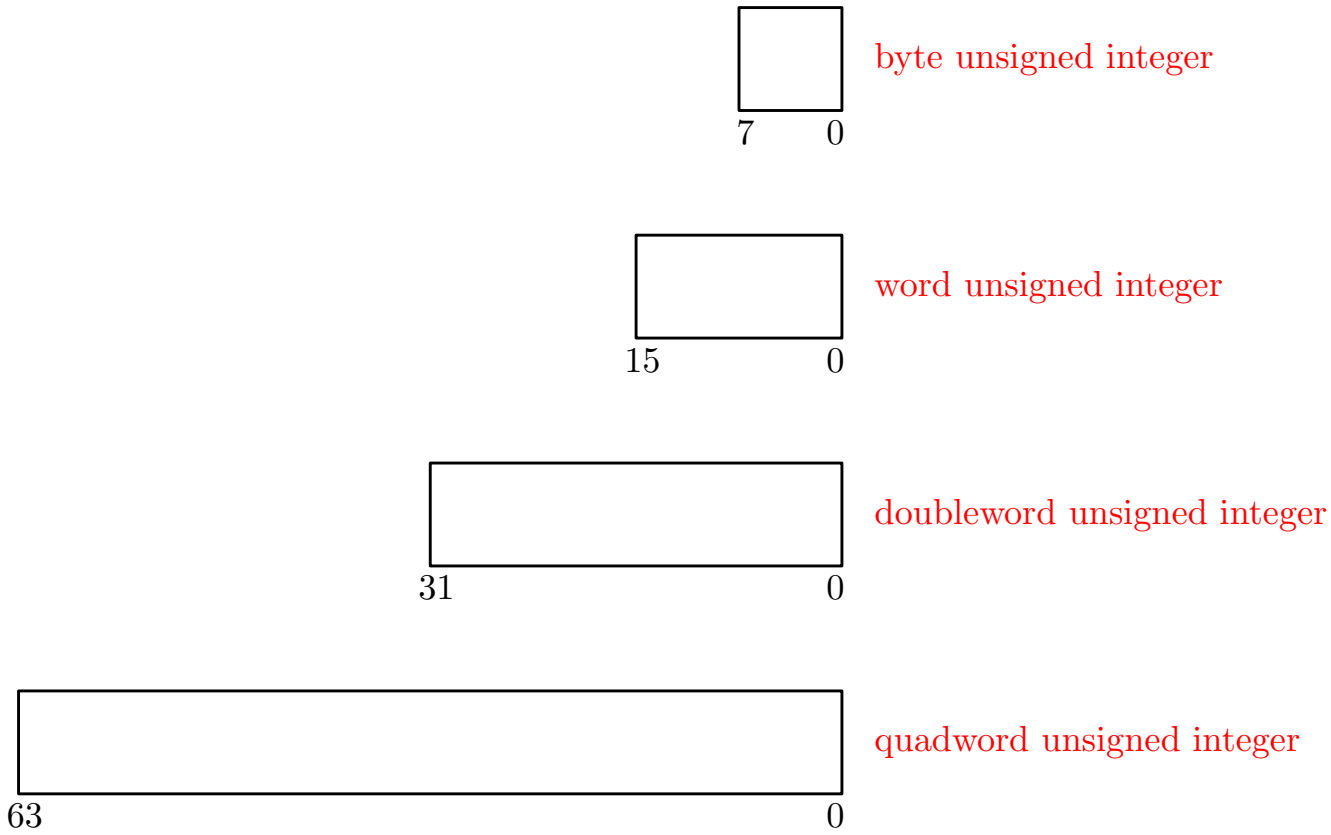
- koji konačni skupovi brojeva modeliraju odgovarajuće matematičke skupove,
- kako se točno prikazuju njihovi elementi u računalu,
- kako se realizira artimetika na tim skupovima.

Detalji u nastavku.

Prije toga, za ilustraciju, pogledajmo kako izgledaju ove tri grupe numeričkih tipova podataka (s podtipovima) na IA–32 arhitekturi.

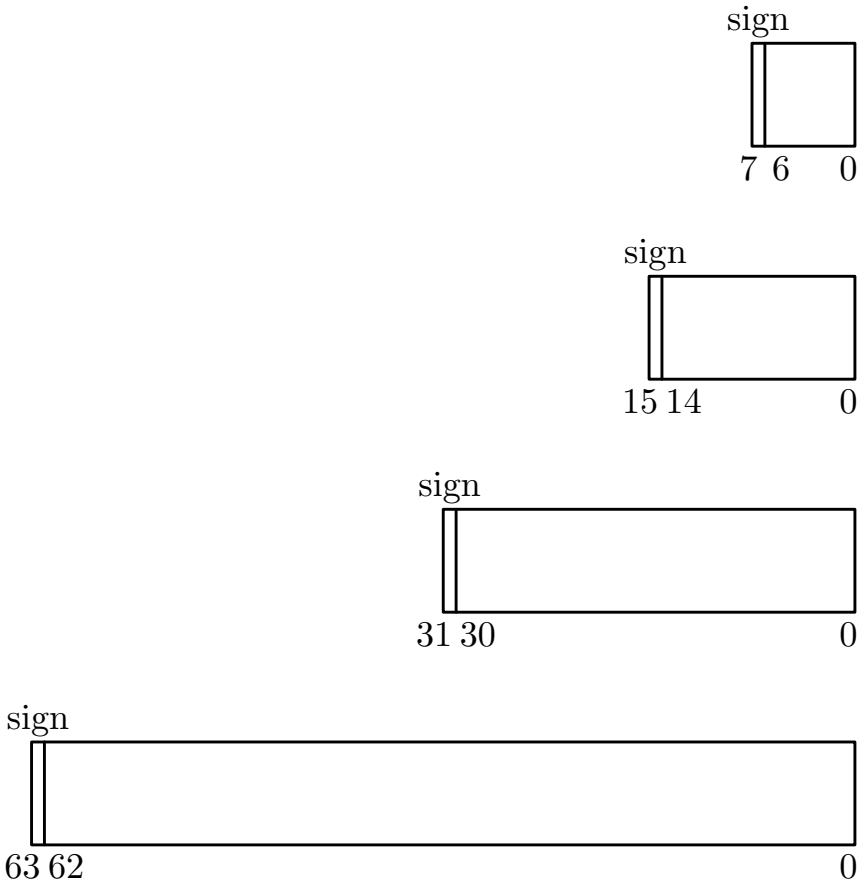
Numerički tipovi podataka na IA–32

Cijeli brojevi bez predznaka (unsigned integer) na IA–32:



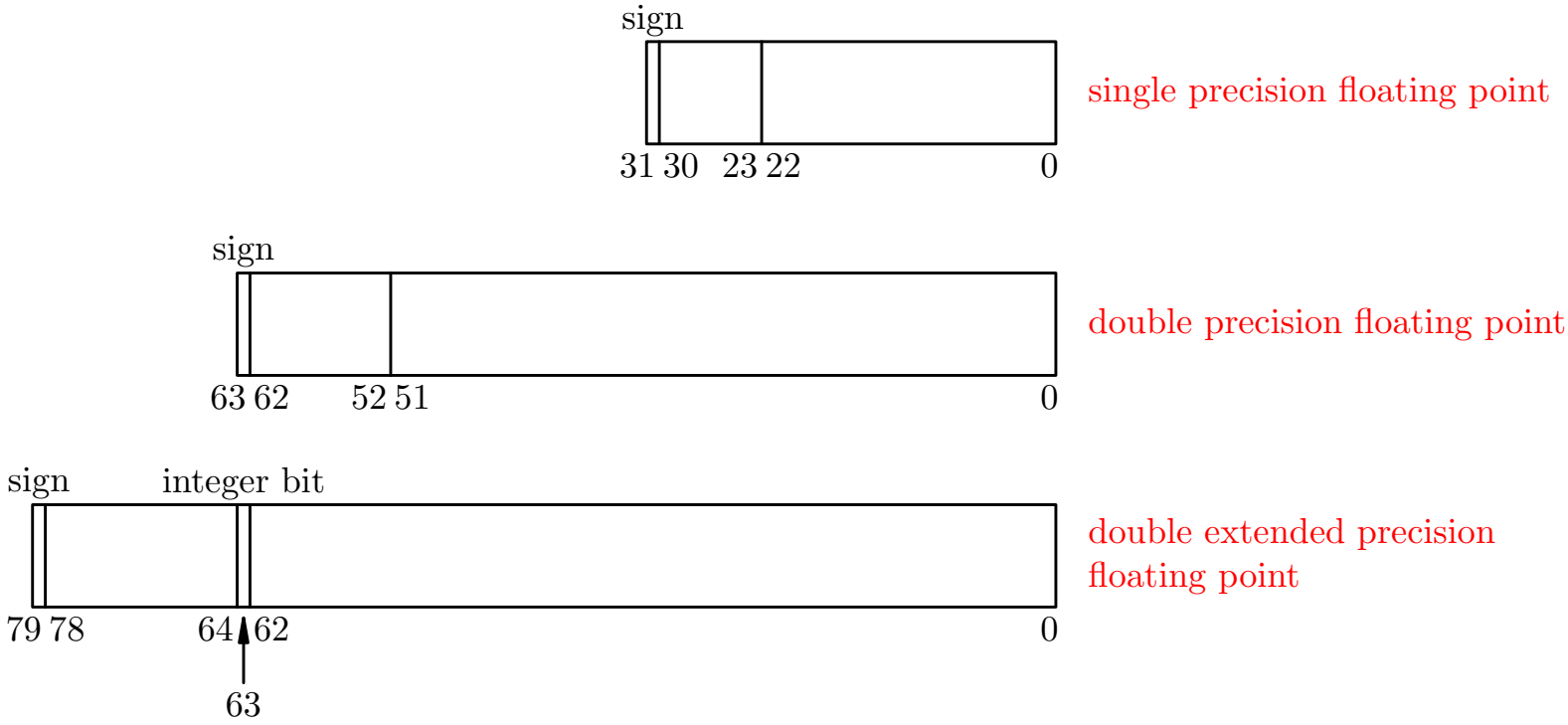
Numerički tipovi podataka na IA–32 (nastavak)

Cijeli brojevi s predznakom (signed integer) na IA–32:



Numerički tipovi podataka na IA–32 (nastavak)

Realni brojevi u tzv. "floating–point" prikazu (v. kasnije), ne samo na IA–32, već općenito, po IEEE standardu:



Skraćeni nazivi za ove tipove su: single, double i extended.

Uvod u prikaz cijelih brojeva

Prvo i osnovno što moramo znati je broj bitova predviđenih za prikaz cijelih brojeva (bez predznaka ili s njim).

Pretpostavimo da imamo n bitova na raspolaganju za prikaz. Već smo vidjeli da su tipične vrijednosti za n

$$8, \quad 16, \quad 32, \quad 64, \quad 128.$$

Danas se (još uvijek) najčešće koristi $n = 32$.

U n bitova možemo prikazati točno 2^n različitih podataka, jer svaki bit može (nezavisno od ostalih) biti jednak 0 ili 1.

Dakle, skup brojeva koje možemo prikazati u tih n bitova ima (najviše) 2^n elemenata. Običaj je da se iskoriste sve mogućnosti za prikaz, pa taj skup ima točno 2^n elemenata.

Cijeli brojevi bez predznaka

Cijeli brojevi bez predznaka modeliraju skup $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$.

Standardni dogovor: u računalu se prikazuje

- najveći mogući početni komad tog skupa $\mathbb{N}_0$.

Ako imamo n bitova na raspolaganju za prikaz, onda skup prikazivih brojeva ima 2^n elemenata, pa je on jednak

$$\{0,\ 1,\ 2,\ \ldots\ ,\ 2^n - 2,\ 2^n - 1\}.$$

Dakle, najveći prikazivi cijeli broj bez predznaka je

$$2^n - 1.$$

Veće brojeve ne možemo prikazati sa samo n bitova.

Cijeli brojevi bez predznaka (nastavak)

Tipične vrijednosti za najveći prikazivi cijeli broj bez predznaka su:

n	$2^n - 1$
8	255
16	65 535
32	4 294 967 295

Kako stvarno izgleda prikaz brojeva u tih n bitova?

Prikaz pojedinih (prikazivih) brojeva je

- doslovna "kopija" prikaza tog broja u pozicionom zapisu u bazi 2.

Što to znači?

Cijeli brojevi bez predznaka (nastavak)

Neka je $B \in \{0, 1, \ldots, 2^n - 1\}$ neki prikazivi broj.

Ako je $B = 0$, onda je svih n bitova u prikazu jednako 0, tj.

$$B = 0 \iff \text{bit}_i = 0, \quad \text{za} \quad i = 0, \ldots, n-1.$$

U protivnom, ako je $B > 0$, onda njegov normalizirani pozicioni prikaz u bazi 2 ima oblik:

$$B = b_k \cdot 2^k + \cdots + b_1 \cdot 2 + b_0, \quad b_i \in \{0, 1\}, \quad b_k > 0,$$

gdje su b_i binarne znamenke (bitovi) broja B.

Ograničenje $b_k > 0$ na vodeću binarnu znamenku samo kaže da je prikaz normaliziran, tj. da nemamo "previše" nul–znamenki "sprijeda".

Cijeli brojevi bez predznaka (nastavak)

Nadalje, znamo da je $k \leq n-1$, jer je B prikaziv.

Ako pišemo samo binarne znamenke (bitove) "u nizu", pripadni pozicioni zapis broja B u bazi 2 ima oblik

$$B = (b_k \; b_{k-1} \; \cdots \; b_1 \; b_0)_2.$$

I točno tako se spremaju bitovi u prikazu, samo treba vodeće bitove dopuniti nulama, od indeksa $k+1$ do $n-1$, ako takvih ima, tj. ako je $k < n-1$.

Dakle, prikaz broja B kao cijelog broja bez predznaka ima oblik

$$\text{bit}_i = \begin{cases} b_i, & \text{za} \quad i = 0, \ldots, k, \\ 0, & \text{za} \quad i = k+1, \ldots, n-1. \end{cases}$$

Cijeli brojevi bez predznaka (nastavak)

U računalu će to biti prikazano kao "cjelina" od n bitova

$$\text{bit}_{n-1}\ \text{bit}_{n-2}\ \ldots\ \text{bit}_1\ \text{bit}_0.$$

Ako "proširimo" zapis broja B u bazi 2 do točno n binarnih znamenki,

$$B = b_{n-1} \cdot 2^{n-1} + \cdots + b_1 \cdot 2 + b_0, \quad b_i \in \{0, 1\},$$

s tim da vodeće znamenke smiju biti 0, odmah dobivamo prikaz broja B kao cijelog broja bez predznaka

$$\text{bit}_i = b_i, \quad \text{za} \quad i = 0, \ldots, n-1,$$

s tim da ovo vrijedi i za $B = 0$.

Cijeli brojevi bez predznaka (nastavak)

Primjer. Uzmimo da je $n = 8$ (da ne pretjeravamo), i pogledajmo zapis broja 123. Za početak, vrijedi

$$123 \leq 255 = 2^8 - 1,$$

pa je 123 prikaziv. Nadalje, njegov binarni prikaz je

$$\begin{aligned} 123 &= 64 + 32 + 16 + 8 + 2 + 1 \\ &= 0 \cdot 2^7 + 1 \cdot 2^6 + 1 \cdot 2^5 + 1 \cdot 2^4 \\ &\quad + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^7 + 1 \cdot 2^0. \end{aligned}$$

Dakle, broj 123 kao cijeli broj bez predznaka ima prikaz

$$123 \quad \longleftrightarrow \quad 0\ 1\ 1\ 1\ 1\ 0\ 1\ 1\ .$$

Aritmetika cijelih brojeva bez predznaka

Aritmetika cijelih brojeva bez predznaka s n bitova za prikaz brojeva je tzv. modularna aritmetika, ili, preciznije

- aritmetika ostataka modulo 2^n.

To znači da aritmetičke operacije $+$, $-$ i $*$ na skupu cijelih brojeva bez predznaka daju rezultat koji je

- jednak ostatku rezultata pripadne cjelobrojne operacije (u skupu $\mathbb{Z}$) pri dijeljenju s 2^n.

Drugim riječima, za prikazive operande A i B vrijedi

$$\text{rezultat } (A \text{ op } B) := (A \text{ op } B) \bmod 2^n,$$

gdje je op zbrajanje, oduzimanje ili množenje.

Aritmetika cijelih brojeva bez predznaka

Važna napomena: Ako je "pravi" cjelobrojni rezultat A op B prevelik (neprikaziv), računalo ne javlja nikakvu grešku, već

- postavlja tzv. bit prijenosa (engl. "carry bit") na 1 u kontrolnom registru.

Što će se dalje dogoditi, ovisi o programu koji se izvršava.

Standardno ponašanje programskih alata ovisi o tome koja vrsta podataka se prikazuje cijelim brojevima bez predznaka. A tu postoje dvije mogućnosti.

- Ako je riječ o "pravim" korisničkim podacima, prijenos se ignorira, bez ikakve poruke. Normalno se nastavlja rad, s rezultatom po opisanom pravilu.

Zato oprez (v. malo kasnije)!

Aritmetika adresa

S druge strane, memorijske adrese se, također, prikazuju kao cijeli brojevi bez predznaka — određene veličine.

Na primjer, obični adresni prostor na IA–32 je 2^{32} byte-a, pa se adrese prikazuju kao 32–bitni cijeli brojevi bez predznaka.

- S adresama se isto tako rade aritmetičke operacije (aritmetika pokazivača ili pointera u C-u), posebno kod obrade nizova.
- Dosta je očito da ova aritmetika nije modularna!
- Tu nema šale, prijenos se ne smije ignorirati i računalo mora javiti grešku ("memory protect violation" ili nešto slično).

Aritmetika adresa (nastavak)

Nažalost, u praksi postoje razne "čarolije" na temu adresa, koje je katkad zgodno znati.

Primjer. Ne znam da li znate da MS Windows XP ima "ograničenje" adresa na samo 2 GB, što je polovina od normalnog adresnog prostora na IA–32 (4 GB).

Stvar ide tako daleko da se XP "ne diže" ako u računalu imate više od 2 GB memorije (probao sam).

U čemu je "štos" — ne znam. Intel kaže da je ograničenje od 2 GB "tvrdo" ugrađeno u osnovne Microsoftove razvojne biblitoeke (tzv. SDK), koje svi proizvođači (pa i Intel) koriste za razvoj svojih programa (recimo, C i Fortran compilera).

Ako netko sazna kako nagovoriti XP da uredno "vidi" više od 2 GB memorije, javite mi.

Aritmetika cijelih brojeva bez predznaka (opet)

Zašto se aritmetika realizira na ovaj način, kao modularna aritmetika?

Postoje dva vrlo dobra razloga.

- Čisto tehnički, ova realizacija je brza.

Ostaci modulo 2^n (uz ignoriranje prijenosa) znače da stalno uzimamo samo najnižih n bitova rezultata, što je lako realizirati.

Drugi razlog je matematičke prirode.

- U pozadini ove realizacije je klasična algebarska struktura prstena ostataka modulo 2^n.

Tu strukturu je korisno detaljnije opisati, jer bitno olakšava razumijevanje cjelobrojne aritmetike (i one s predznakom).

Prsten ostataka modulo 2^n

Naime, skup svih prikazivih cijelih brojeva bez predznaka

$$\mathbb{Z}_{2^n} = \{0,\ 1,\ 2,\ \ldots\ ,\ 2^n - 2,\ 2^n - 1\}$$

je ujedno i standardni sustav ostataka koji dobivamo pri cjelobrojnom dijeljenju s 2^n. Zato se i označava sa $\mathbb{Z}_{2^n}$.

Ako na njemu definiramo binarne operacije zbrajanja $\oplus$ i množenja $\odot$ preko ostataka pripadnih cjelobrojnih operacija,

$$A \oplus B := (A + B) \bmod 2^n,$$
$$A \odot B := (A \cdot B) \bmod 2^n,$$

onda $(\mathbb{Z}_{2^n}, \oplus, \odot)$ ima algebarsku strukturu prstena.

Prsten ostataka modulo 2^n (nastavak)

Lako se vidi da za (jedinstveni) suprotni element "$-A$" obzirom na zbrajanje vrijedi: "-0" $= 0$, i "$-A$" $= 2^n - A$, za $A \neq 0$, jer je

$$A \oplus \text{“}-A\text{”} = (A + (2^n - A)) \bmod 2^n = 0.$$

Na kraju, oduzimanje $\ominus$ definiramo kao zbrajanje sa suprotnim elementom

$$A \ominus B := A + \text{“}-B\text{”}.$$

I tako smo dobili tri osnovne aritmetičke operacije, koje se upravo na taj način realiziraju u računalu za cijele brojeve bez predznaka.

A što je s dijeljenjem? To dosad nismo ni spomenuli!

Prsten ostataka modulo 2^n (nastavak)