Komandas izpilde un datora veiktspēja

Komandu kopa

Augsta līmeņa valodas kods : C, C++, Java, Fortan,

kompilators

Asamblera valodas kods: arhitektūras specifiskas komandas

asamblers

Mašīnkods: arhitektūrai specifikas bitu virknes

programmatūra

Komandu kopa

aparatūra

Datora sastāvdaļas

- Neatkarīgi no paaudzes katrā datorā var izdalīt šādas galvenās sastāvdaļas:
  - aritmētiski loģisko mezglu;
  - vadības mezglu;
  - atmiņu;
  - Ievada / izvada iekārtas.

Datora sastāvdaļas - ALM

- Aritmētiski loģiskais mezgls - summators
- Karodziņi – komandas rezultāta loģiskais novērtējumu.

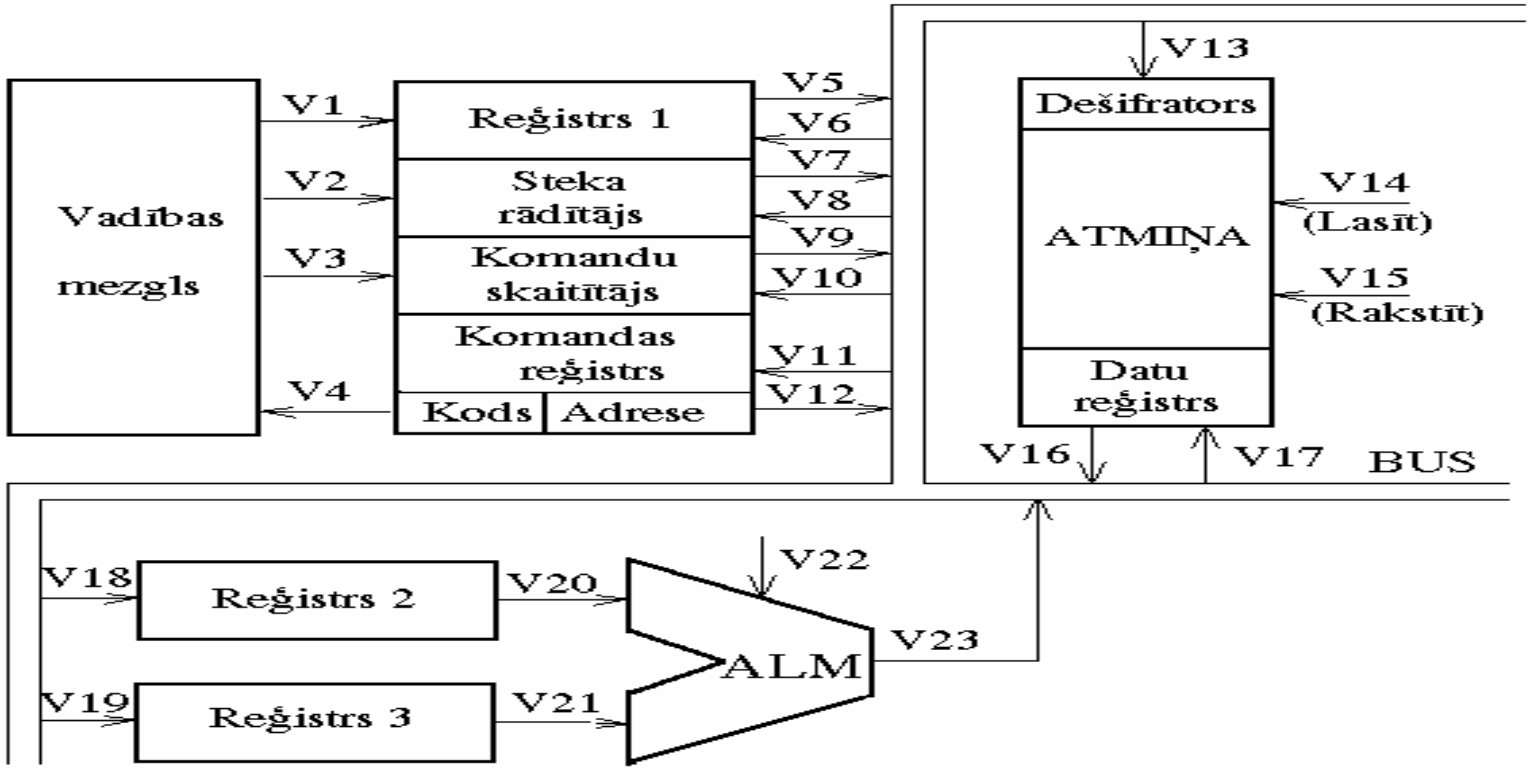
Datora sastāvdaļas - ALM

- Reģistri parasti no 8 līdz 256 vārdiem
  - plaši pielietojami: pagaidu vērtības un mainīgie
  - reģistri jānosauc kādā vārdā
- Keši vs. Reģistri
  - Reģistri
    - Ātrāki (nav adresācijas režīmu, tegu)
    - Determinēta piekļuve “deterministic” (nav piekļuves kļūdu)
    - Var tikt replicēti ar domu nodrošināt vairākas vienlaicīgas piekļuves
    - Īss identifikators
    - Jāpieglabā/jāatjauno apakšprogrammu izsaukumu gadījumā
    - Nevar iegūt reģistra adresi (atšķirībā no atmiņas)
    - Fiksēts apjoms
    - Kompilatoram tie jāpārvalda
- Cik reģistru vajag? Vairāk == labāk?
  - Var ilgāk uzglabāt operandus (izpildes laiks un atmiņas datu plūsma samazinās)
  - Garāki identifikatori (izņemot gadījumu kad tiek lietoti reģistru logi)
  - Vairāk konteksta palēnina konteksta pārslēgšanas laiku

Datora sastāvdaļas – vadības mezgls

- Vadības mezgls nodrošina vadības signālu izstrādāšanu, kuri nepieciešami visu datora bloku saskaņotai darbībai laikā.
  - komandu skaitītājs - norāda pašreiz izpildāmās komandas numuru jeb adresi
  - komandas reģistrs - komandas kods
  - karodziņu reģistrs - palīdz vadīt turpmāko programmas izpildīšanas virzienu.

Datora mezgli

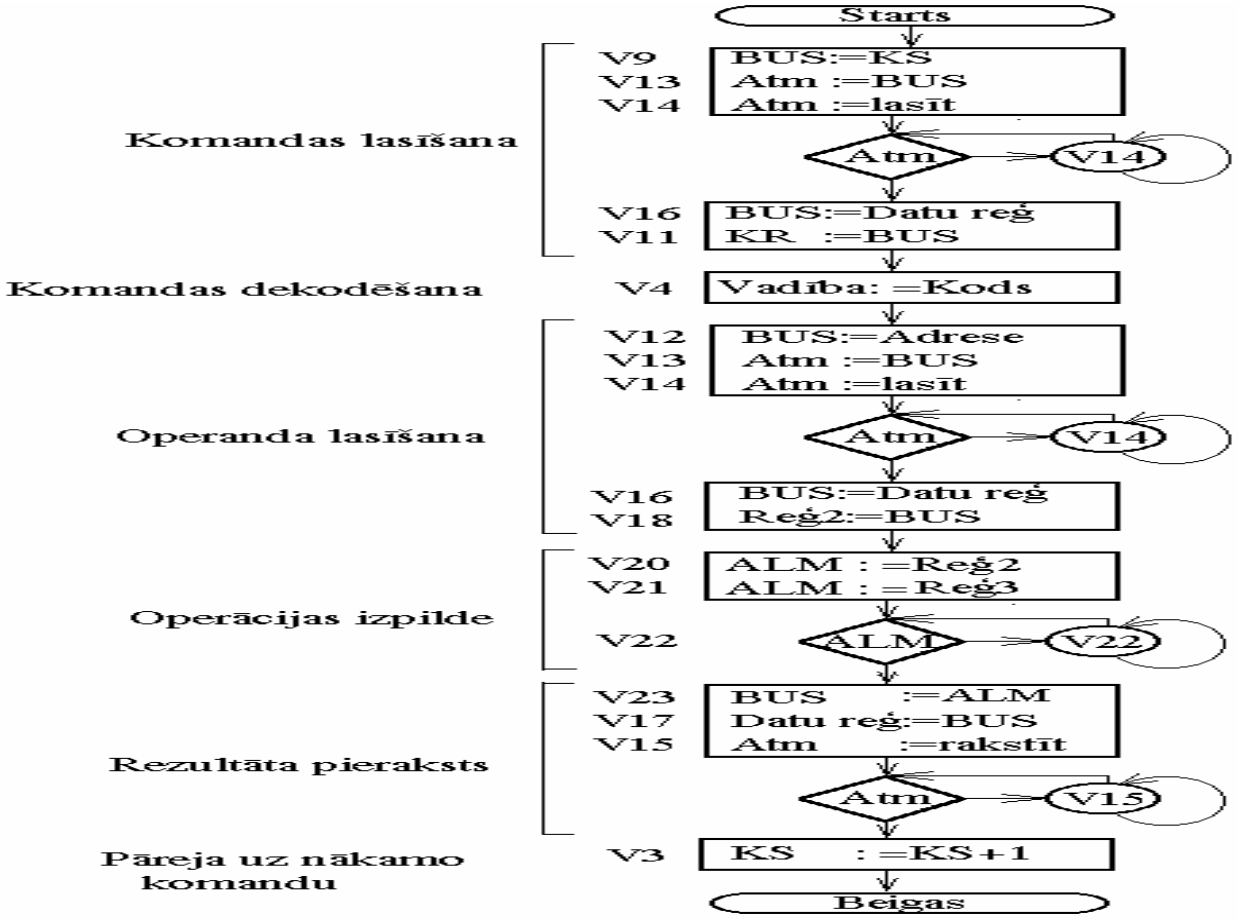


3.1. zīm. Klasiska vienkārša datora galvenie mezgli un vadības signāli

Komandas izpilde

- Katras komandas izpildīšanas laiku var sadalīt sīkākos cits citam sekojošos laika posmos:
  - komandas nolasīšana no atmiņas un ievietošana komandas reģistrā;
  - komandas izpildīšanai nepieciešamo vadības signālu veidošana;
  - operandu nolasīšana no atmiņas un ievietošana ALM darba reģistros;
  - komandā prasītās darbības izpildīšana;
  - pāreja uz nākamās komandas izpildīšanu.

Komandas izpilde



3.2. zīm. Vienkāršota komandas izpildes secība

Vadības nodošanas komandas

- Vadības nodošanas komandas ļauj automātiski vadīt skaitļošanas procesu un izpildīt programmu zarošanos.
- To ir trīs veidi:
  - nenosacītā vadības nodošana;
  - nosacītā vadības nodošana;
  - apakšprogrammu izsaukšana un atgriešanās no apakšprogrammām.

Datorsistēmu veiktspēja un mērījumi

- Parasti mūs interesē uzdevuma izpildes laiks
- **Uzdevuma izpildes laiks (sek./uzd.) v.s caurlaides spēja (uzd. / sek.)**
- Kādi tad ir uzdevumi:
  - Reālās programmas (vispārējas nozīmes, zinātniskās...)
  - Etalonuzdevumi
    - Pārstāvošās programmas (SPEC, SYSMARK, u.t.t.)
    - Kodoli - koda fragmenti (Linpack)
    - Spēļu programmas (Quicksort, Sieve of Eratosthenes)
    - Sintētiskās programmas – mix. (Whetsone, Dhrystone)
- Veiktspēja – veikto darbu daudzums laika vienībā
- N-reizes ātrāk

$$n = \frac{\text{Veiktspēja (X)}}{\text{Veiktspēja(Y)}} = \frac{\text{Izpildes laiks(Y)}}{\text{Izpildes laiks(X)}}$$

Veiktspējas mērīšana

- Parasti PC gadījumā normalizē izpildes laikus uz etalondatora un tad aprēķinā vidējo vērtību normalizētajām izpildes laika vērtībām (*aritmētisko* vai *ģeometrisko* vidējo vērtību)
- Aritmētiski vidējā vērtība mainās ar etalondatora nomaiņu un to iespaido “izlecēji”. Varētu likt svara koeficientus bet KĀ tos likt ir ?
- Ģeometriskā vidējā vērtība nemainās ar etalondatora nomaiņu bet....
  - Piem. ir divas programmas kas ir testa kopā kur 1. izpildās 2. sek. bet otra 10. sek. Ja uzalabo kādas izpildi 2x tad rezultāts ir vienāds?
  - Bez tam rezultāts ir ***attiecība*** nevis laiks.
- Ideālā gadījumā
  - Mērām reālu uzdevumu kopas un liekam svaru koeficientus pēc tā cik bieži patiesībā tiks izpildīta tā vai cita programma
  - Ja tas nav iespējams tad vismaz jānovienādo izpildes laiki uz kāda etalondatora un jāpaziņo šie svara koeficienti
  - Jānodefinē dati ar kuriem tiek veikti testi

Vēl kādas vidējās vērtības?

- Harmoniskā vid. vērtība?
  - –Piem. braucot pusi ceļa ar 40 Km/h un otru pusi ar 90 Km/h. Kāds ir vidējais ātrums?
  - –(40 + 90)/2=65 Km/h
  - –(40*90)^0.5=60 Km/h
  - –2/((1/40)+(1/90))=54 Km/h
  - –Pareizi būtu kopējais ceļš/laiku. Piem 10 Km kopējam ceļam tas ir:
    - 10/((5/40) + (5/90)) =
    - Nav svarīgi cik garš ir ceļš?

Metrikas - MIPS

- MIPS = komandu skaits / (izpildes laiks x 10^6) =
  = takts frekvence / (CPI x 10^6)
- Problēmas:
  - Komandu kopas nav vienādas (RISC v.s CISC)
    - 1 CISC komanda > 1. RISC komanda
  - Programmas sastāv no dažādu komandu sajaukumiem
  - Ir izmantojams ja salīdzina vienādus etalonuzdevumus, vienādas komandu kopas, vienādus kompilatorus un OS

Metrikas - MFLOPS

- Līdzīgi MIPS, tikai skatoties uz FP komandām
  - Tāpat nav nekam noderīgs reālajā dzīvē
  - FP darbību izmantojošas lietotnes
- Tradicionāli FP darbības bija lēnas un pārtraukumus varēja neņemt vērā bet:
  - šodien atmiņa var būt šaurā vieta
  - “Peak MFLOPS” ir marketings un ne vairāk jo pamatā tas ir:
    izpildes mezglu skaits x takts frekvence

Metrikas - GHz

- Cik laba ir šāda metrika? Tāda pat kā pārējās?
  - Viena vērtība, nekādu testu, lielāks == labāks... Ideāli?
  - Diemžēl ir realizācijas ir ar domu vinnēt šo “karu”.
- Piem. SPEC FP2000 rezultāti:
  - 833 2.4 GHz Intel Pentium 4
  - 1356 1.0 GHz Itanium-2
  - 456 1.4 GHz Intel PIII
  - 434 450 MHz IBM POWER3

CPU veiktspēja

CPU laiks	=	Sekundes / Programu	=	Komandas / Programā	x	Taktis / Komandai	x	Sekundes / Taktij

- 500 MHz Pentium III procesoram lai izpildītu lietotni ar 200K komandām ir nepieciešamas 2 ms.
- 300 MHz UltraSparc procesoram tai pašai lietotnei (kura dotajā komandu kopā satur 230K komandas) ir nepieciešamas 1.8 ms.
- Kāds ir CPI katram procesoram dotajai programmai?
  - CPI = Taktis / Komandu skaitu = CPU laiks X Takts frekvence / Komandu skaits
  - $CPI_{Pentium} = 2*10^{-3}$ X $500*10^{6}$ / $2*10^{5}$ = 5.00
  - $CPI_{SPARC} = 1.8*10^{-3}$ X $300*10^{6}$ / $2.3*10^{5}$ = 2.35
- Kurš ir ātrāks un par cik?
  - UltraSparc ir 2/1.8 = 1.11reizes ātrāks, jeb par 11% ātrāks.

Etalonuzdevumu pamata kļūdas

  - Tiek parādīta tikai kāda vidējā vērtība
  - Kešošanas efekti tiek ignorēti
  - Tiek lietoti neatbilstoši buferu izmēri
  - Tiek ignorēts monitoringa iespaids
  - Netiek nodrošinātas vienāds sākuma stāvoklis
  - Savāc daudz datus bet neizdara analīzi
- Krāpšanās veidi
  - Veido speciālus kompilatorus dotajai darba uzdevumu kopai
  - Lieto ļoti mazus etalonuzdevumus
  - Manuāli translē etalonuzdevumus lai optimizētu to veiktspēju

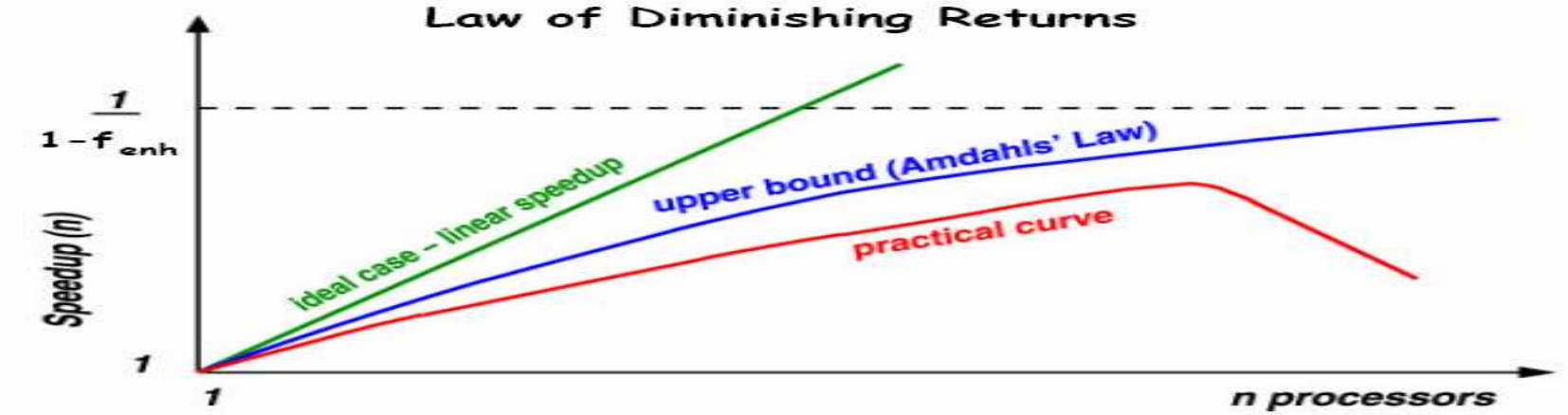
Kā uzlabo veiktspēju

- Ātrākas tehnoloģijas bet:
  - Izmaksas aug
  - Uzticamība krītas
  - 3.10^8 m/sek
- Lielākas matricas (SOC - System On a Chip)
  - Mazāk vadu bet zemāks iznākums IC dēļ kļūdām
- Paralēlā skaitļošana nCPU
  - Vai var sagaidīt S = n ?
- Konveijerapstrāde

Amdala likums

- Amdala likums apgalvo ka ja ir kāda programmas daļa kuru var optimizēt tad kopējais ieguvums ir izsakāms kā:

  Kopējais uzlabojums=1/((1-P)+(P/S))
- Piemērs:
  - ir 10% programmas koda kuru var izmest (S=inf.) tad labākais rezultāts būs: 1/(1-0.1)=1.11111...reizes
  - Ir 90% programmas koda kuru var uzlabot par 20% un tad rezultāts būs: 1/ ((1-0.9)+(0.9/1.2))=1.1746...reizes



Noslēgums

- DA cenšas uzlabot bieži izmantojamās lietas
- Vislabāk veiktspēju ļauj novērtēt reālas lietotnes vai to būtību atainojoši etalonuzdevumi (benchmarks).
- Izpildes laiku nosaka komandu skaits lietotnē, vidējais komandu skaits vienā taktī un takts laiks
- Veidojot datorsistēmas jāņem vērā gan cena gan veiktspēja

Mājas darbi

- Izlasīt:
  - http://en.wikipedia.org/wiki/Benchmark_%28computing%29
  - http://en.wikipedia.org/wiki/Arithmetic_mean
  - http://en.wikipedia.org/wiki/Geometric_mean
  - http://en.wikipedia.org/wiki/Harmonic_mean
  - http://dt.cs.rtu.lv/viewfile.php/18/file/4/434/3.clekcija.pdf
- Piedāvāt savu nenosacītās vadības komandas realizāciju 3.1 zīmējumā dotajā datorā
- Kāds būs uzlabojums gadījumā ja var samazināt programmas komandu skaitu divas reizes?
- Miniet ātrākas tehnoloģijas izmantošanas piemēru veiktspējas uzlabošanai.
- Miniet etalonuzdevumu kopu kurai ir pieļaujama uzdevumu manuāla translācija.