

Ievads Datoru Arhitektūrā

Pasniedzējs: Aigars Riekstiņš

Kursa lapa: <http://ida.cs.rtu.lv>

E-pasts: aigars@ida.cs.rtu.lv

Konsultācijas: M1/4 319 kab. Ceturtdienās 14:30 – 18:05



Kursa apskats

- Lekcijas 1x. nedēļā
- Praktiskie darbi - 1x. divās nedēļās
- Apmeklējums gan lekcijās gan pr. nodarbībās ir obligāts un tas tiek reģistrēts
- Gala un vidus semestra pārbaude – eksāmens (rakstiski bez palīgīdzekļiem)
- Eksāmenu rezultāti (katrs) veido 40% no gala vērtējuma atzīmes. Eksāmena vērtējumi pēc 10 ballu sistēmas tiek ieskaitīts gala vērtējumā pēc šādas skalas:
 - eksāmena vērtējums no 1 līdz 3 dod 0 balles gala vērtējumā,
 - eksāmena vērtējums no 4 līdz 5 dod 2 balles gala vērtējumā,
 - eksāmena vērtējums no 6 līdz 8 dod 3 balles gala vērtējumā,
 - eksāmena vērtējums no 9 līdz 10 dod 4 balles gala vērtējumā.
- Eksāmenam pielaisti tiek tikai tie studenti kuriem ir studentu apliecības (derīgas)
- Nav vienas literatūras vienības kuru izlasot var veiksmīgi nokārtot kursu
- Jautājumi un iebildumi tiek iedrošināti gan tiešā formā (lekcijās) gan anonīmi vai nesaistes režīmā forumos, e-pastos ...

Kursa lapa

- Lai veiksmīgi sadarbotos šajā kursa lapā studentiem jāveic reģistrācija un jāpiereģistrējas sava m.g. kursa apakšlapā:

*First Name:	<input type="text"/>
*Last Name:	<input type="text"/>
*Email Address:	<input type="text"/>
*Username:	<input type="text"/> ?
*Password:	<input type="text"/> ?
*Confirm Password:	<input type="text"/>
<input type="button" value="Add"/> <input type="button" value="Cancel"/>	

1.kurss	2. kurss
 Members <input type="button" value="Join"/>	 Members <input type="button" value="Join"/>

*First Name:	Jānis
*Last Name:	Bērziņš
Stud. apl. Nr.:	RDB*****
*Email Address:	janis.berzins@*****
*Username:	students ?
*Password:	<input type="text"/> ?
*Confirm Password:	<input type="text"/>
Preferred Language:	English <input type="button" value="v"/>

Aptuvenais kursa saturs

- **Vispār:**
 - dot priekšstatu par datora arhitektūru no programmētāja viedokļa
 - dot salīdzinošu priekšstatu par dažādām arhitektūrām
- **Šodien:**
 - iepazīšanās ar kursa saturu, darba veidu, vērtējuma veidu un saziņas iespēju izskaidrošana
 - arhitektūras sastāvdaļas;
 - arhitektūras pamata definīcijas;
- Pārējais <http://ida.cs.rtu.lv>

Kāpēc jāapgūst datoru arhitektūru?

- Lai varētu vismaz izprast procesoru un sistēmu projektēšanas principus
- Lai spētu pamatoti pieņemt lēmumus par sistēmas konfigurāciju un kompromisiem tās veidojot. (cik lielam kešam/atmiņai jābūt, kādas kopnes izmantot savienojot komponentes, kādus un cik diskus jālieto.....)
- Lai spētu izvēlēties datoru konkrētam darba uzdevumam
- Lai spētu izprast tirgotāju skaisto slaidu saturu (etalonuzdevumu rezultātus u.t.t.)
- Lai spētu izvēlēties procesora veidu konkrētam uzdevumam
- Lai spētu veidot sistēmas sw (OS, kompilatorus)
- Lai apgūtu vairāku arhitektūru assemblerus
- Lai kļūtu par procesora izstrādes grupas vadītāju ?

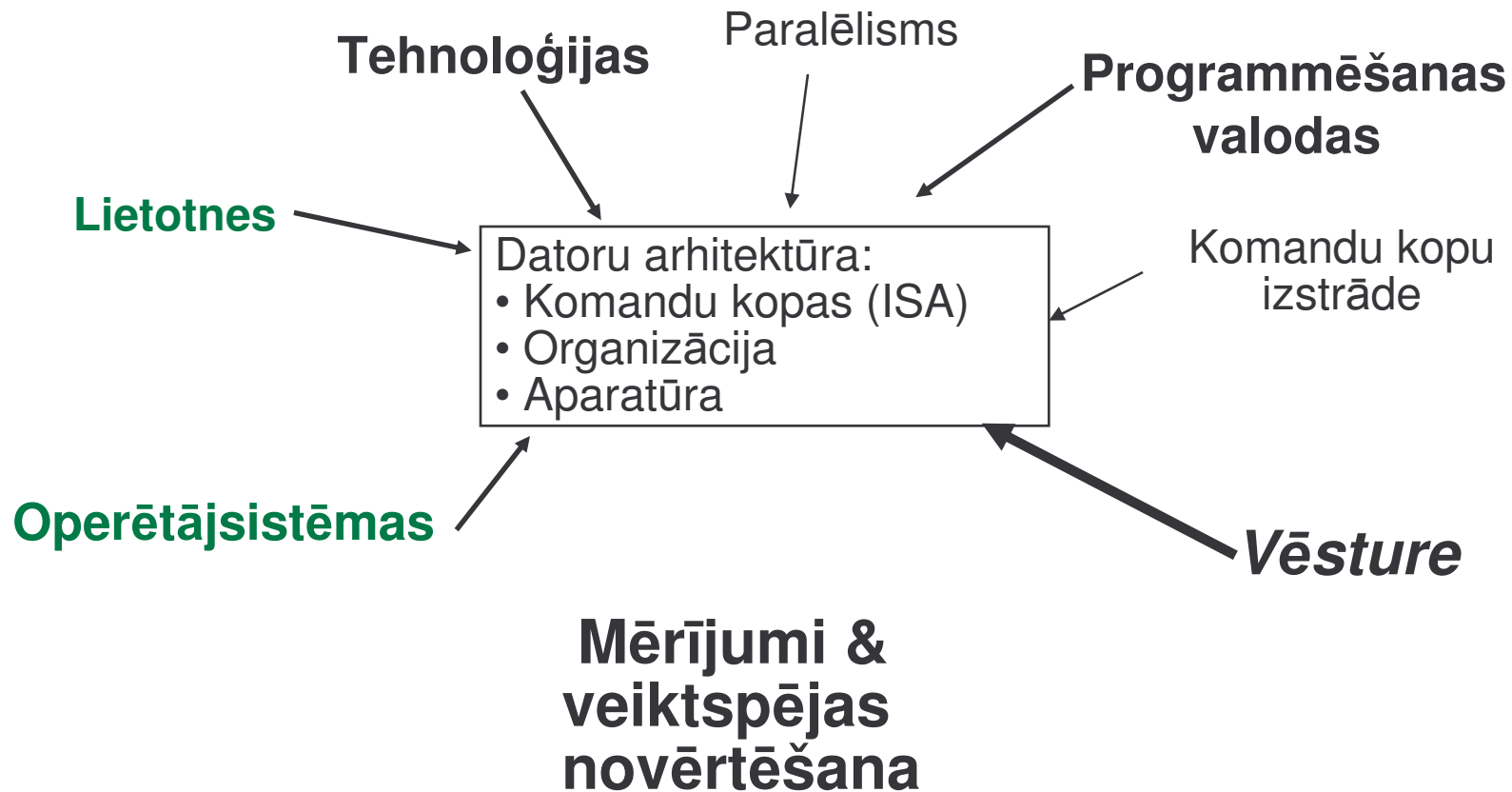
Kas tad ir DA

- Datoru Arhitektūras termins radies IBM kad Amdahl, Blaauw, and Brooks [1964] definēja to kā programmētājam redzamo komandu kopu. Tika uzskatīts ka datori ar vienādu arhitektūru varēs izpildīt programmas bez izmaiņām tajās.
- Pozitīvais:
 - Precīzi definētas arhitektūras var tikt realizētas dažādos veidos
 - Programmas kas rakstītas vienādās komandu kopās var tik izpildītas visās savietojamās realizācijās
- Mūsdienās datoru arhitektūra tiek definēta kā:
 - Komandu kopas arhitektūra (ISA) + Organizācija (konveijerizācija, atmiņas hierarhija, uzglabāšanas sistēmas ...)
 - ISA piemēri:
 - Digital Alpha (v1, v3) 1992
 - HP PA-RISC (v1.1, v2.0) 1986
 - Sun Sparc (v8, v9) 1987
 - MIPS (MIPS I, II, III, IV, V) 1986
 - Intel (8086,80286,80386,80486,Pentium, MMX, ...) 1978
 - Itanium 2001
 - Cell 2005

Arhitektūra ⇔ Realizācija

- Vienai arhitektūrai var būt vairākas realizācijas
 - Datoru saimes
- Vairākas arhitektūras var tikt izveidotas izmantojot vienu realizāciju
 - Mikrokode emulatori

IDA kurss



Prasības pret DA

- Lietotnes
 - Vispārējas nozīmes
sabalansēta veikspēja dažādiem uzdevumu apgabaliem (cena / veikspēja)
 - Zinātniskās
augsta peldošā punkta aritmētikas veikspēja un adresējamās atmiņas apjoms
 - Iebūvētās
Zema cena un patērētā jauda (min. nepieciešmā jauda uzdevumam)
 - Komerčiālās
decimālās aritmētikas atbalsts, datubāzu/transakciju apstrāde (veiktspēja, pieejamība, mērogojamība)
- SW saderības veidi
 - Objektkoda / binārā līmenī
nav nepieciešama sw pārnesamība, lielāka hw izstrādes izmaksas
 - Programmēšanas valodas līmenī
atrisina veco arhitektūru nastu

Prasības pret DA

- Operētājsistēmu prasības
 - Adrešu lauks
 - Atmiņas pārvaldība / aizsardzība
 - Reāllaika darbu plānošana
 - Pārtraukumi / slazdi (traps)
- Standarti
 - Peldošā punkta (IEEE754)
 - I/O kopņu
 - OS
 - Tīklu
 - Programmēšanas valodu

Uz ko balstās DA

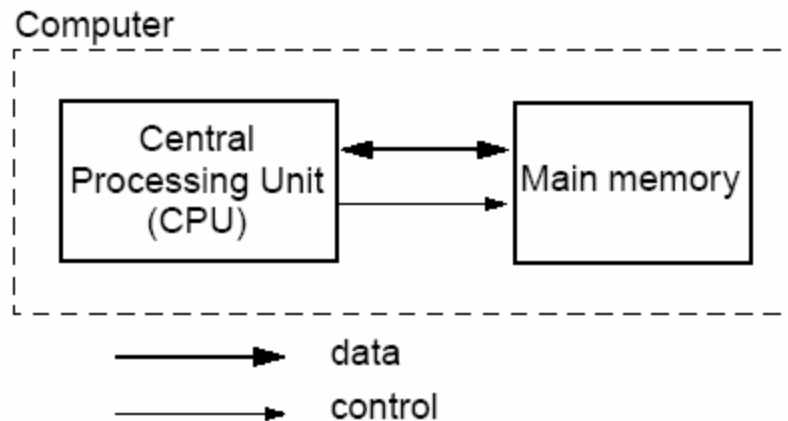
- skaitļu teorija
- skaitļojošā matemātika
- galīgo automātu teorija
- Petrī tīkli
- algoritmu teorija
- rindošanas teorija
- varbūtību teorija
- drošuma teorija
- elektrotehnika, shēmtehnika
- siltumtehnika.....

Ko tad saprot ar vārdiem “Dators” un “Datorsistēmas”

- ?

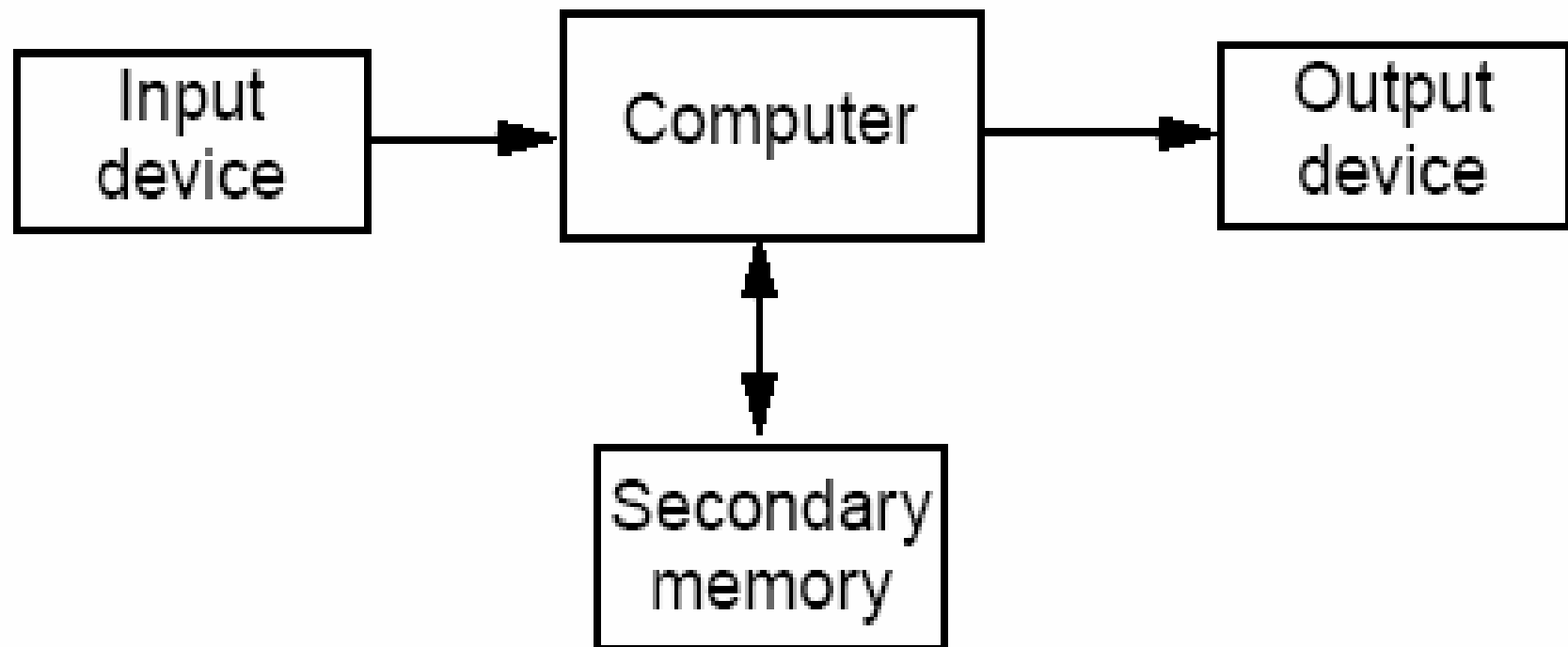
Dators un datorsistēmas

- Kas tad ir dators?
 - Tehniska sistēma (ierīču komplekts), kas saskaņā ar uzdotu programmu veic matemātisku datu apstrādi.



- Un datorsistēma?
 - Datora un tā perifērijas ierīču (t. sk. diskdziņu, monitora, dažādu ievadizvades ierīču u. c.) pilna konfigurācija, kas operētājsistēmas vadībā kopīgi veic datu apstrādi un ievadizvadi.

Datori un datorsistēmas



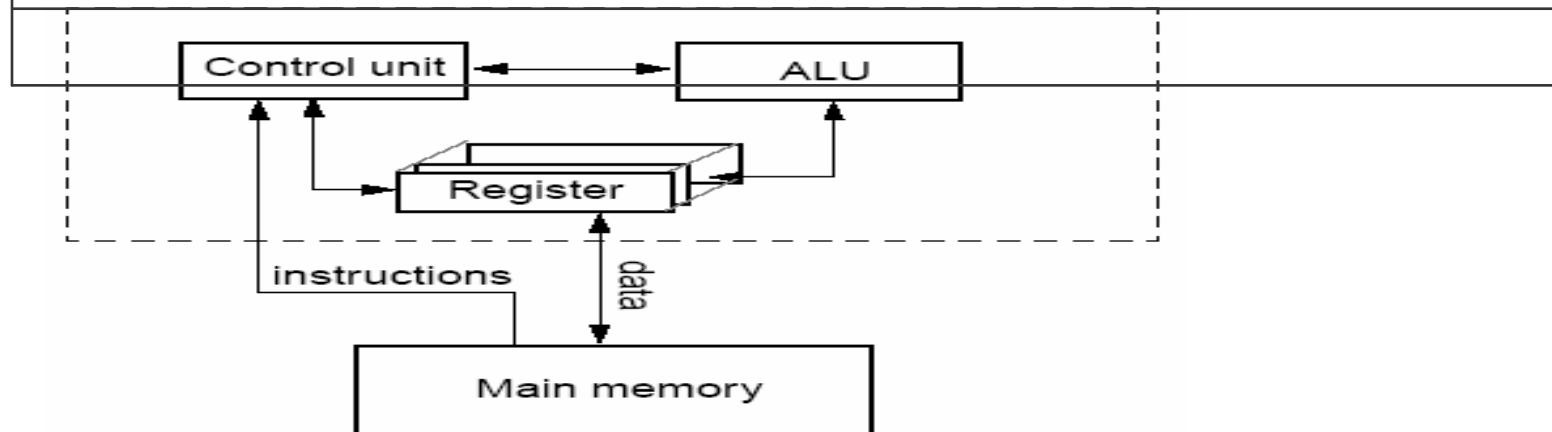
von Neumann arhitektūra

- Pamatprincipi:
 - Dati un komandas tiek glabāti pamatatmiņā
 - Pamatatmiņa ir adresējam pēc vietas (neatkarīgi no tā kas atrodas dotajā vietā)
 - Komandas tiek izpildītas secīgi (pēc kārtas kādā tās atrodas atmiņā) ja izpildes secība netiek speciāli mainīta
 - Datora organizācija:
 - Centrālais procesors (CPU) kas satur vadības mežglu kurš koordinē visu komandu izpildi un aritmētiski loģisko mežglu (ALU) kurš savukārt izpilda prasītās komandas
 - Pamatatmiņa
 - Von Neumann datori ir vispārējas nozīmes datori (tas ir tie var veikt ļoti dažādus uzdevumus atkarībā no programmas ko tie izpilda)

von Neumann arhitektūra

Vispārējas nozīmes (von Neumana) arhitektūras:

- Aparatūra veic ļoti dažādus uzdevumus atkarībā no tā kāda programma tiek izpildīta
- CPU uzdevums ir izpildīt komandas kas tiek saņemtas no atmiņas
- Komandas liek procesoram veikt kādu no pamata darbībām (aritmētiskās, loģiskās, datu pārvietošanas...)
- Vadības mezgls ir tas kurš atpazīst (dekodē) komandas un vada citu komponentu darbību
- CPU satur pagaidu uzglabāšanas vietas kurās parasti uzglabā bieži izmantotos datus un rezultātus



Harvardas arhitektūra

- Arhitektūra kurā dati un komandas atrodas fiziski dažādās atmiņās
- Ir iespējams lasīt nākamo komandu kamēr pieraksta iepriekšējās rezultātu
- Pamatā šodien ir atrodama visos plaša pielietojuma CPU un ciparu apstrādes shēmās (DSP), iegultajos kontroleros (PIC)

Datu attēlojums

- Datorā dati un vadības informācija tiek attēloti binārajā formā kurā eksistē tikai divi simboli “0” un “1”

- Šie simboli attēlojas kā elektriskie signāli.

- Skaitļi tiek attēloti kā 2. pakāpes:

$$100101 = 1 \cdot 2^0 + 0 \cdot 2^1 + 1 \cdot 2^2 + 0 \cdot 2^3 + 0 \cdot 2^4 + 1 \cdot 2^5$$

$$10110 = 0 \cdot 2^0 + 1 \cdot 2^1 + 1 \cdot 2^2 + 0 \cdot 2^3 + 1 \cdot 2^4$$

- Binārie kodi tiek apstrādāti tieši (tos nepārveido 10. sistēmā):

$$100101 + 10110 = 111011$$

Programmas izpilde

- Piemēram

$Z := (Y + X) * 3$

- Katra komanda tiek izpildīta kā soļu kopa. Visi soļi kas attiecināmi uz vienu komandu tiek saukti par komandas ciklu (*instruction cycle*).

Address	
00001000	<div>0000101110001011</div> <div>Move addr of Y Reg 3</div>
00001001	<div>0001101110000011</div> <div>Add addr of X Reg 3</div>
00001010	<div>0010100000011011</div> <div>Mul operand "3" Reg 3</div>
00001011	<div>0001001110010011</div> <div>Move addr of Z Reg 3</div>
.....
01110000	00000000000001011 ← X
01110001	00000000000000011 ← Y
01110010	00000000000101010 ← Z

Q&A

BUJ

Moore's law

