Latvijas Universitāte
Datorikas fakultāte

CPU. DataPath

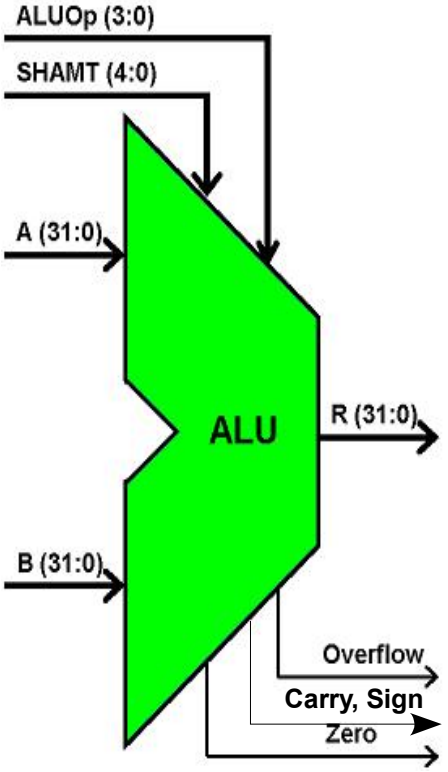
Kurss "ievads digitālajā projektēšanā"
Lekcija 05.10.2012

Autors: Rinalds Ruskuls

Aritmētiskais-loģiskais mezgls (**A**rithmetic-**L**ogic **U**nit)

ALU pamatoperācijas

- Loģiskās (AND, OR, XOR, NOT)
- Aritmētiskās (“+”, “-”, “·”, “/”)
- Bitu nobīdes operācijas
- Salīdzināšanas operācijas.



MIPS

- **M**icroprocessor without **I**nterclocked **P**ipeline **S**tages
- Pieder pie ***RISC*** (**R**educed **I**nstruction **S**et **C**omputing)
- Eksistē 32 un 64 bitu versijas (šajā kursā runāsim par 32 bitu versiju)
- Šāda tipa mikroprocesori tiek izmantoti rūteros (Cisco), spēļu konsolēs (PlayStation 1&2), kā arī citās iegultās sistēmās

Vientakts CPU implementācija

- Lekcijā tiks izstāstīta vienkārša *MIPS CPU* arhitektūras implementācija, kas atbalsta šādas operācijas:
  - Aritmētiskās
  - Datu ierakstīšana / nolasīšana
- Šodienas lekcijā tiks apskatīts vientakts MIPS CPU
  - Katra operācija izpildās *vienā* taktī
  - Uzsvars – datu maģistrāle

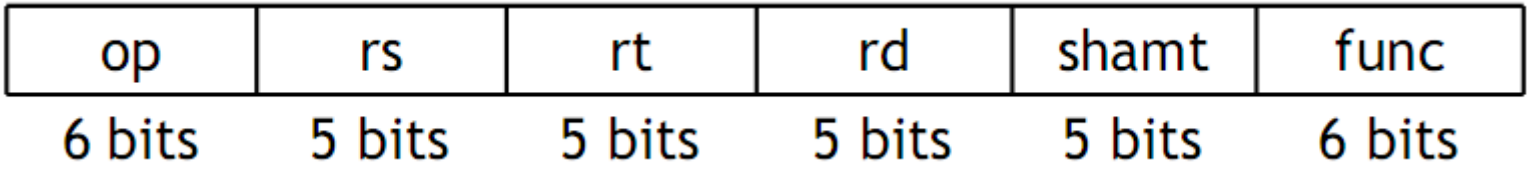
MIPS instrukciju tipi

Type	-31-		format (bits)			-0-
R	opcode (6)	rs (5)	rt (5)	rd (5)	shamt (5)	funct (6)
I	opcode (6)	rs (5)	rt (5)	immediate (16)		
J	opcode (6)	address (26)				

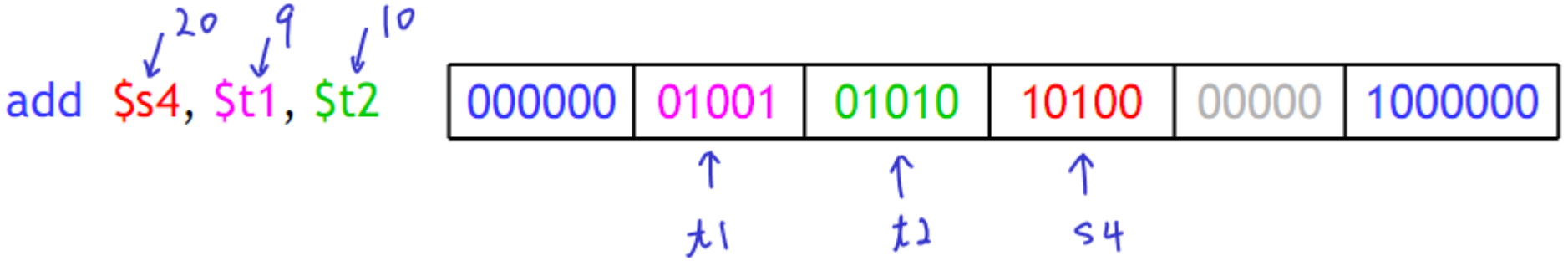
- Opcode – darbības tips
- rs,rt – ieejas reģistri, rd – izejas reģistrs
- Shamt – nobīdes vērtība
- FUNCT – darbības kods
- Immediate, addres – skaitliskās vērtības

R komandu izpildīšana

- Divu reģistru aritmētiskām darbībām izmanto ***R*** tipa instrukciju tipu
  - ***Op*** – instrukcijas opkods, ***func*** – nosaka izpildāmo aritmētisko operāciju
  - ***Rs***, ***rt***, ***rd*** ieejas, izejas reģistri

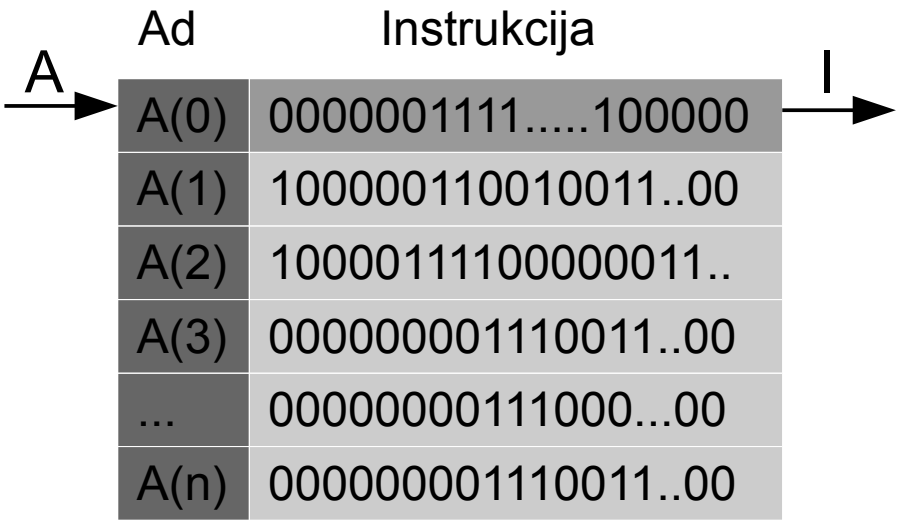


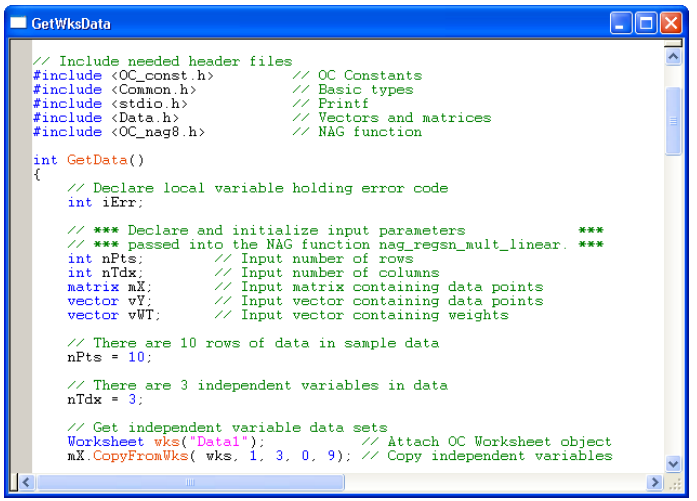
- Instrukcijas un tās atkodēšanas piemērs



Instrukciju atmiņa (Instruction Memory)

- Glabā programmu (instrukciju)
- Atgriež instrukciju pēc uzdotās adreses satura
- Vairums gadījumos nepieciešams tikai lasīšanas režīms:
  - DataPath neveic izmaiņas programmā
  - Programmas ievade atmiņā ir uzskatāma par specifiku gadījumu un var tikt realizēta atsevišķi



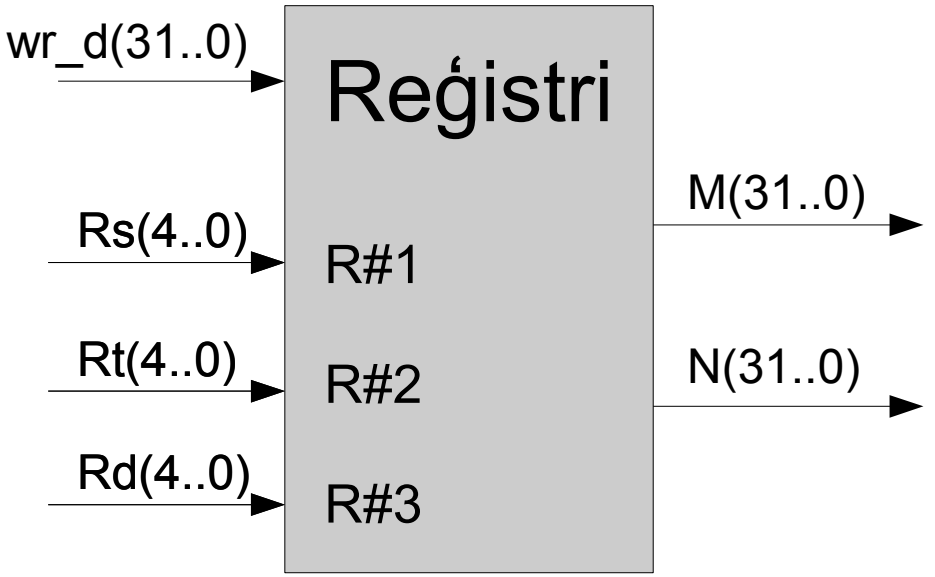


Reģistru fails (Register file)

Reģistri – atmiņa, kurā iespējams rakstīt un lasīt

Lasīt ir iespējams divu reģistru vērtības vienlaicīgi – ***M***, ***N***

Rakstīt vienu vērtību - ***wr_d***



Mūsu izmantotais reģistrs var saglabāt 32 - 32 bitu vērtības

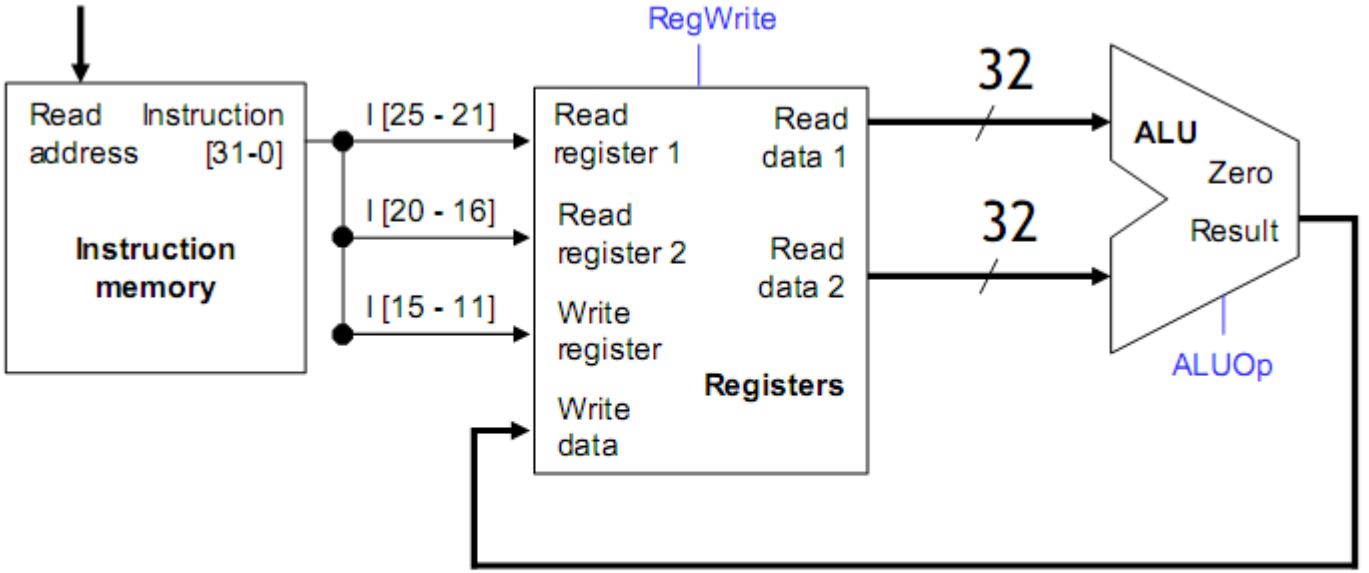
- ***rs***, ***rt***, ***rd*** – definē reģistra kārtas numuru

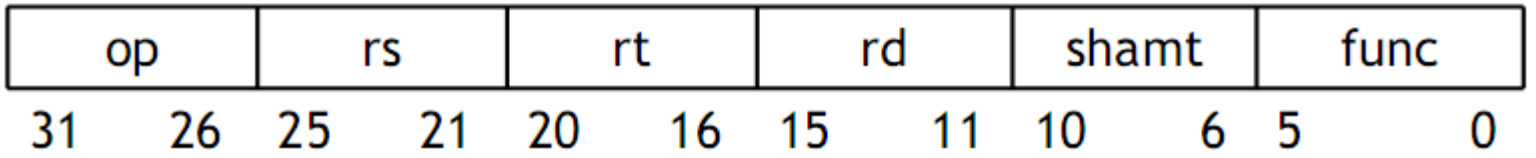
1) Nolasa instrukciju no instrukciju atmiņas – padodot adresi

2) Definētie reģistri (***rs***, ***rt***) tiek nolasīti no reģistru faila

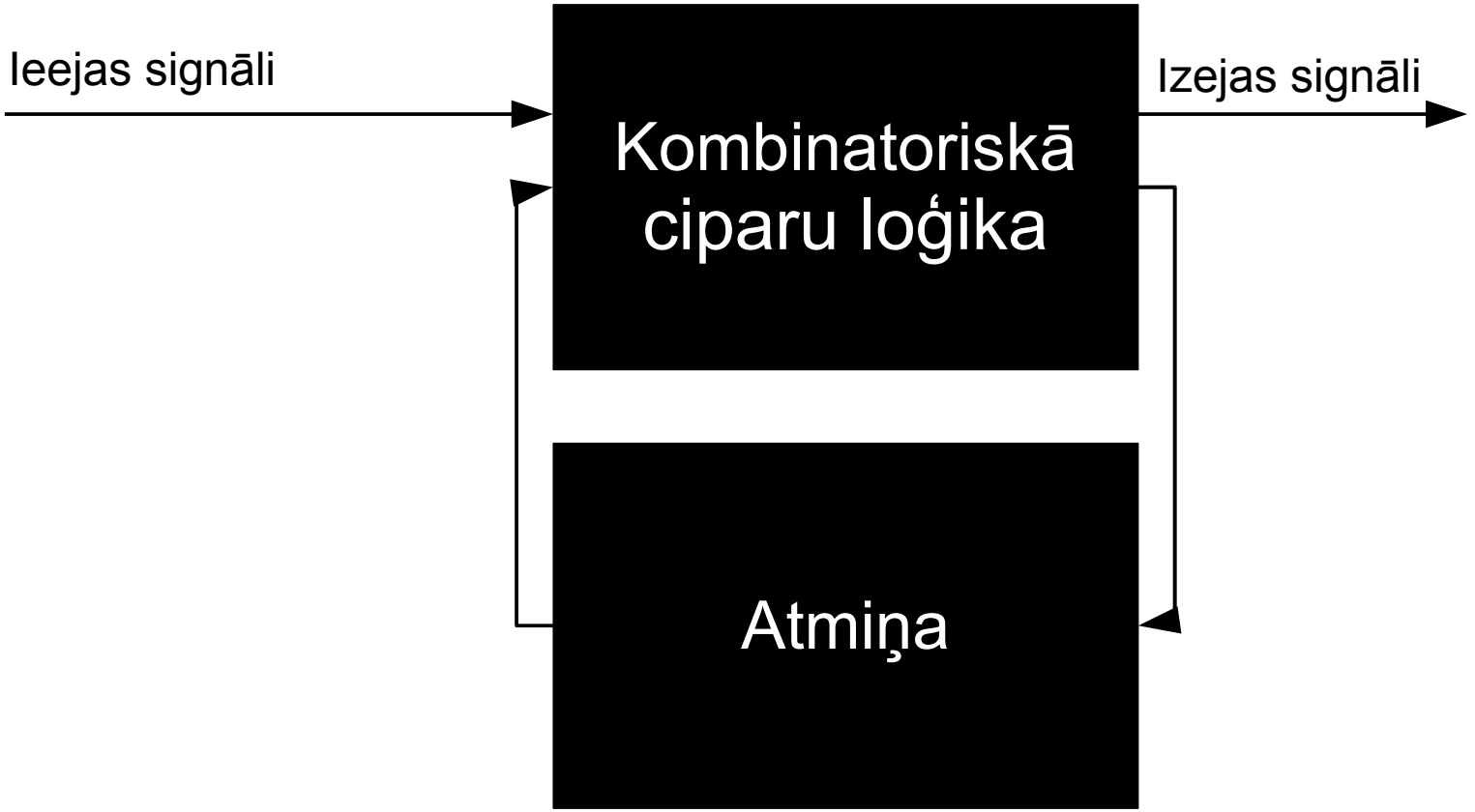
3) ALU – izpilda aritmētisko operāciju

4) Rezultāts tiek salgabāts izejas reģistrā ***rd***





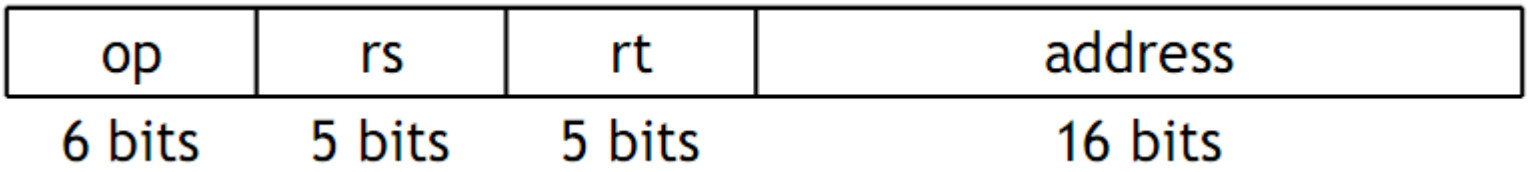
Ir līdzības ar iepriekšējo shēmu?

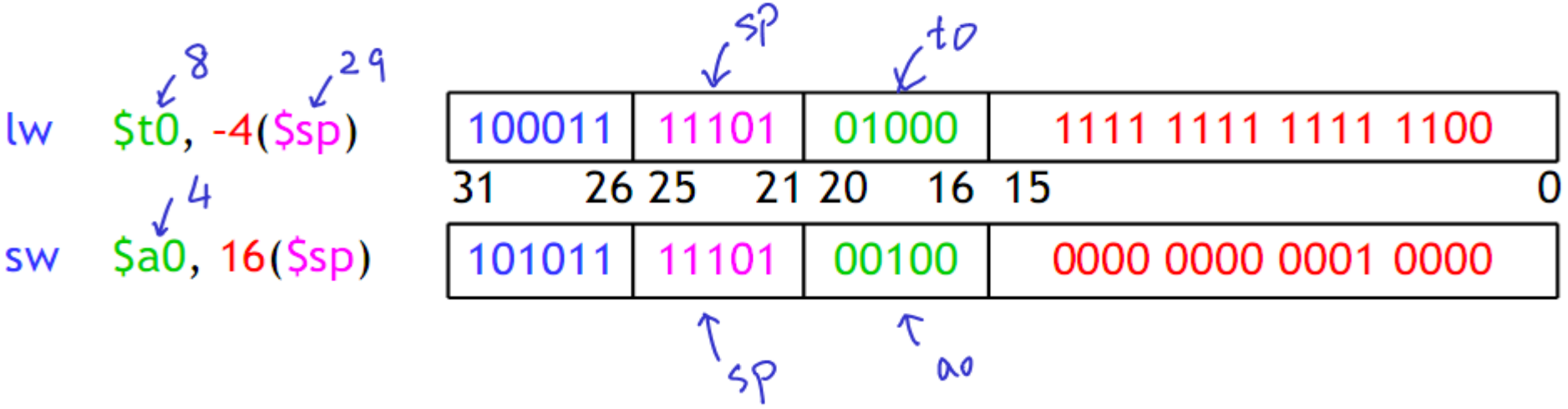


I komandu izpildīšana

Komandas kā ***lw***, ***sw***, ***beq izmanto I*** tipa kodējumu

- ***rt*** ir izejas reģistrs priekš ***lw***, bet priekš ***sw***, ***beq*** ieejas
- ***address*** – 16 bitu konstante



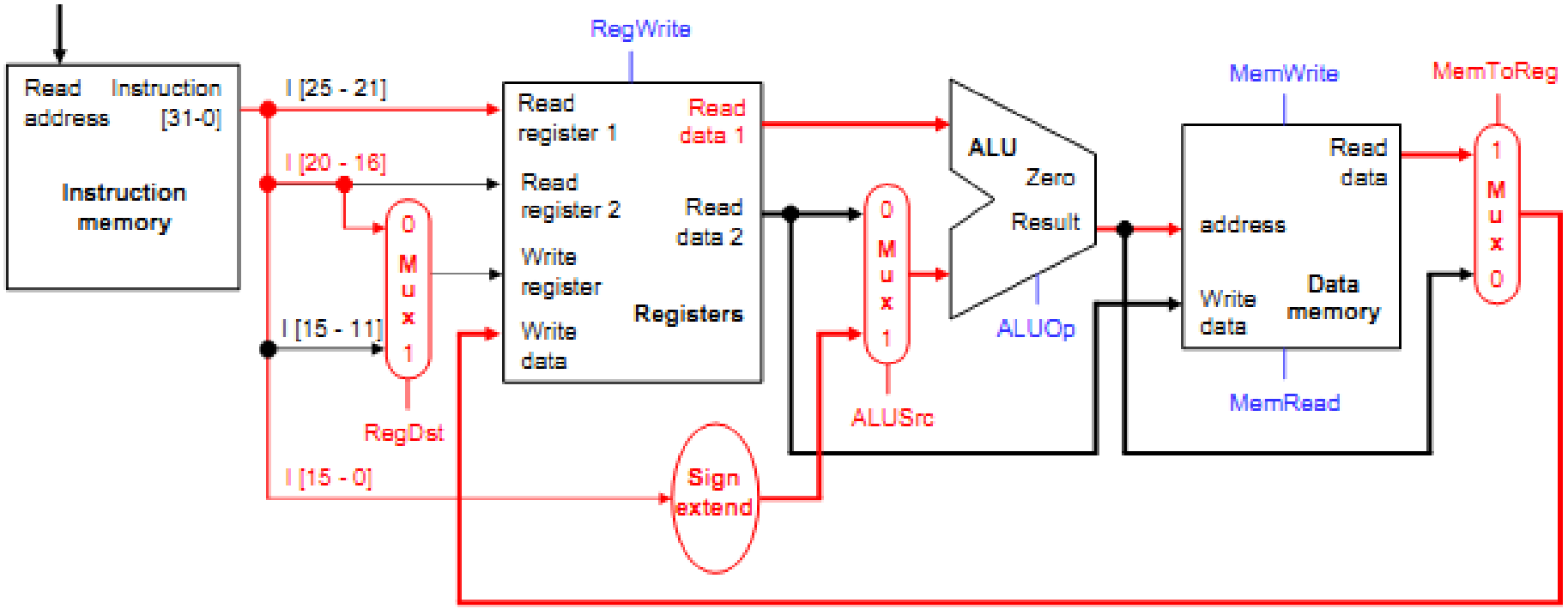


Shēmas kopskats

Ja nepieciešams izpildīt šādu operāciju – lw $t0, -4($sp), bāzes reģistrs $sp ir saskaitīts ar ***sign-extendet*** konstanti, adreses vērtības iegūšanai

Nozīmē, ka ALU jāspēj saņemt gan reģistru vērtības, gan ***sign-extendet immediate*** operandus priekš ***lw*** un ***ws*** komandām

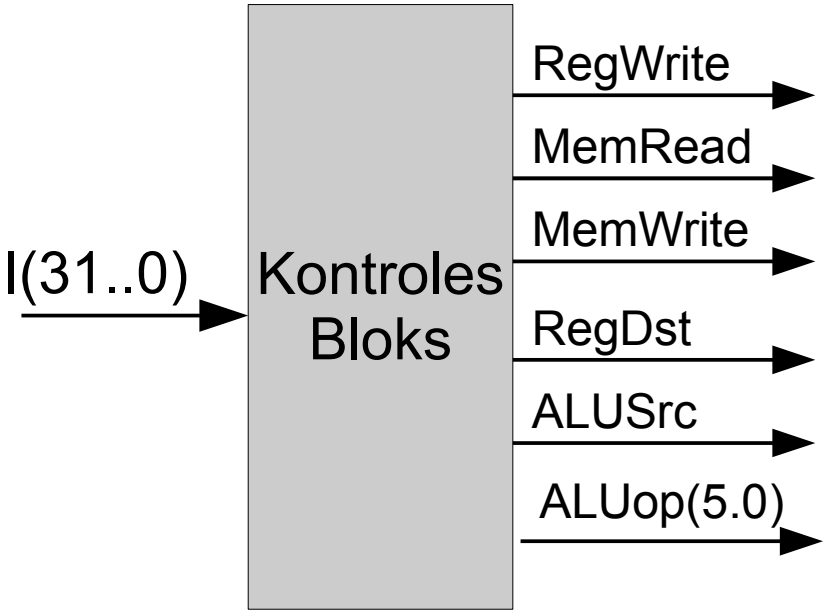
Ievietojot multipleksoru – tiek panākta vēlamā funkcionalitāte



Kontroles bloks

Shēmas daļa, kas rūpējas par to, lai katra operācija izpildītos pareizi

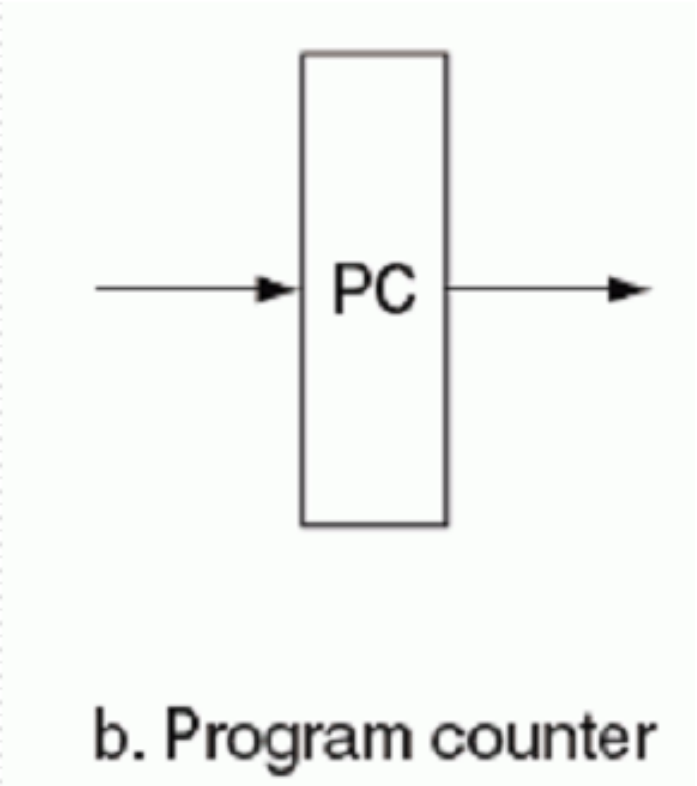
- Ieejā ir instrukcija – bitu virkne (32 bitu gara)
- Izejā signāli, kas kontrolē multipleksorus, ALU utt (iepriekšējās shēmās attēloti zilā un sarkanā krāsā)



Vairums signālu tiek ģenerēti izejot no opkoda vērtības

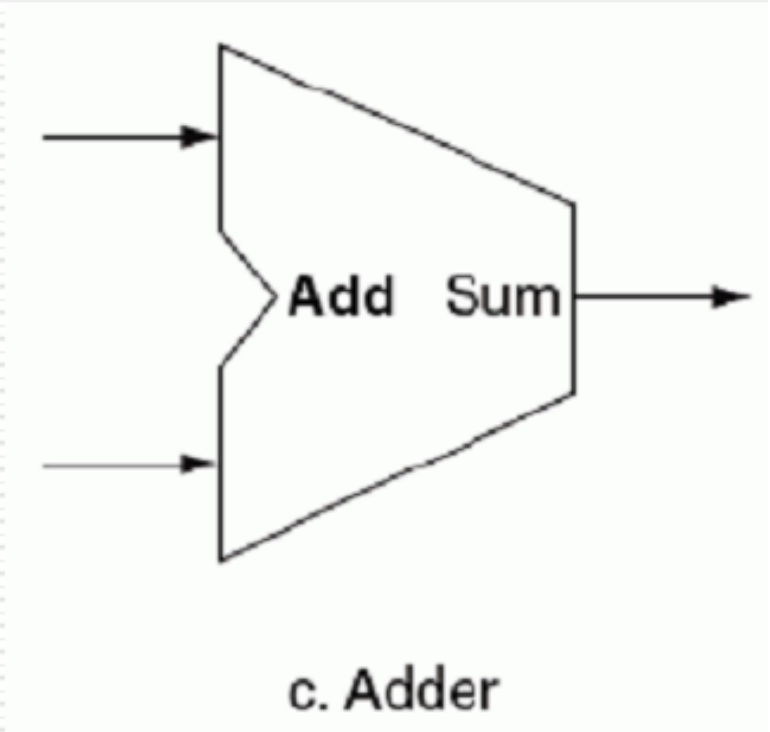
Programm counter (PC)

- 32 bitu reģistrs
- Satur aktuālās instrukcijas adresi
- Tiek pārrakstīts pēc katras operācijas

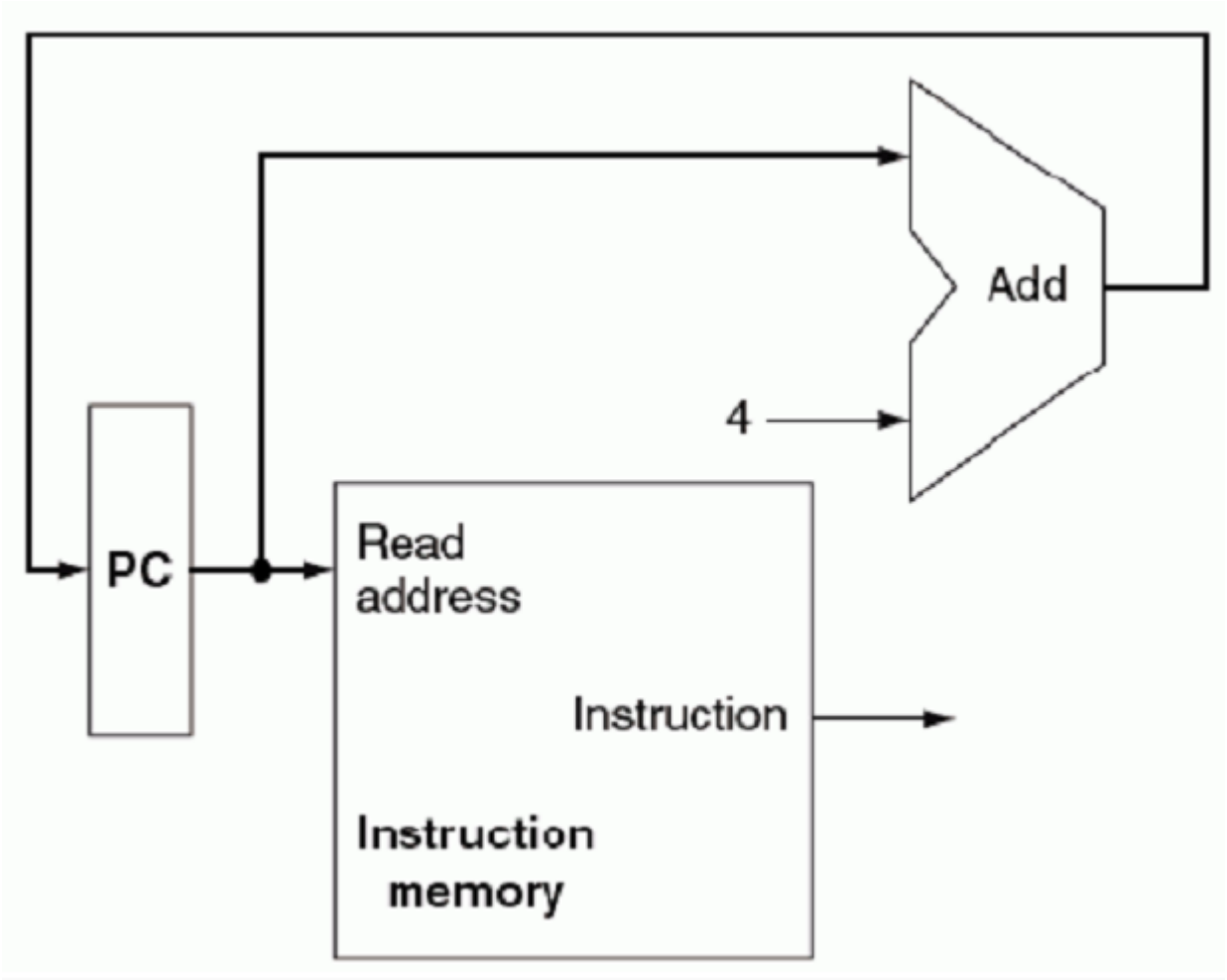


Skaitītājs (Adder)

- Nepieciešams, lai izrēķinātu nākošās instrukcijas adresi
- Skaitīšanas konstante ir atkarīga no instrukcijas garuma



Apskatītie moduļi kopskatā



Paldies par uzmanību!
Jautājumi?