Arhitektura računara

dr.sc. Amer Hasanović

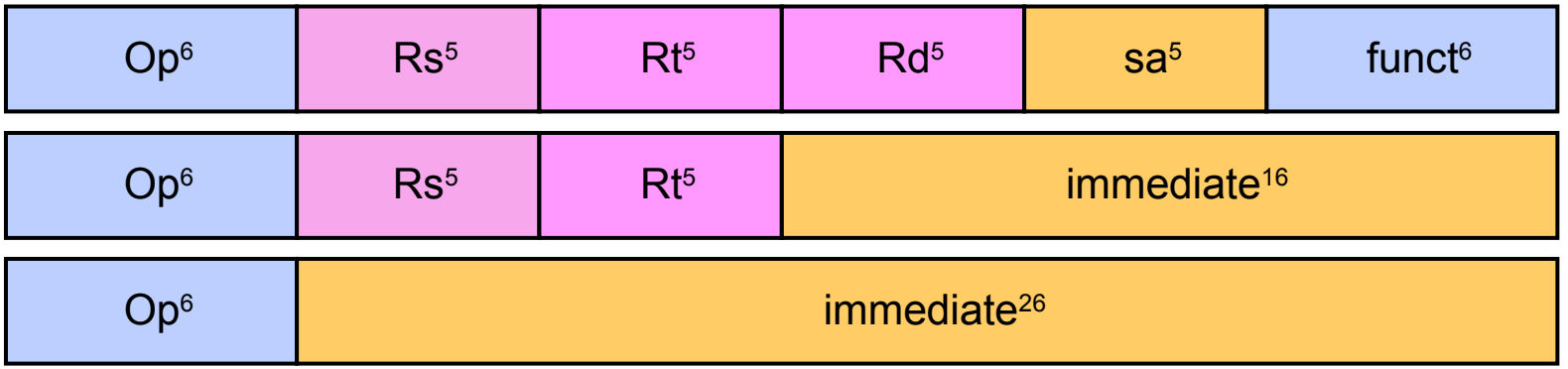
Pregled

- Princip rada CPU
  - Staza podataka (datapath)
  - Jednociklusna implementacija

U predavanju korišteni segmenti iz prezentacije autora M. Mudawar, PhD: http://faculty.kfupm.edu.sa/coe/mudawar/

MIPS instrukcije kodiranje

- Instrukcije su kodirane u 32 bita
- Koriste se tri formata:



- Op^6: 6-bita opcode polje
- Rs^5, Rt^5, Rd^5: 5-bita izvorni i destinacijski registri
- sa^5: 5-bita iznos za šift operaciju
- $funct^6$: 6-bita polje koje se koristi za određivanje funkcije R tipa instrukcije
- $immediate^{16}$: 16-bita ofset u adresi ili konstanta
- $immediate^{26}$: 26-bita destinacijska adresa skok instrukcije

Izvršavanje instrukcija

- **R-tip**

  Dohvati instrukciju: Instruction ← MEM[PC]
  Pročitaj operande: data1 ← Reg(Rs), data2 ← Reg(Rt)
  Izvrši operaciju: ALU_result ← funct(data1, data2)
  Zapiši rezultat: Reg(Rd) ← ALU_result
  PC ← PC + 4

- **I-tip**

  Dohvati instrukciju: Instruction ← MEM[PC]
  Pročitaj operande: data1 ← Reg(Rs), data2 ← Extend(imm16)
  Izvrši operaciju: ALU_result ← op(data1, data2)
  Zapiši rezultat: Reg(Rt) ← ALU_result
  PC ← PC + 4

- **BEQ**

  Dohvati instrukciju: Instruction ← MEM[PC]
  Pročitaj operande: data1 ← Reg(Rs), data2 ← Reg(Rt)
  Izvrši oduzimanje: zero ← subtract(data1, data2)
  Izvrši grananje: if (zero) PC ← PC + 4 + 4×sign_ext(imm16)
  else PC ← PC + 4

Izvršavanje instrukcija

- **LW**

Dohvati instrukciju:	Instruction ← MEM[PC]
Pročitaj bazni reg.:	base ← Reg(Rs)
Sračunaj adresu:	address ← base + sign_extend(imm16)
Pročitaj memoriju:	data ← MEM[address]
Zapiši rezultat:	Reg(Rt) ← data
	PC ← PC + 4

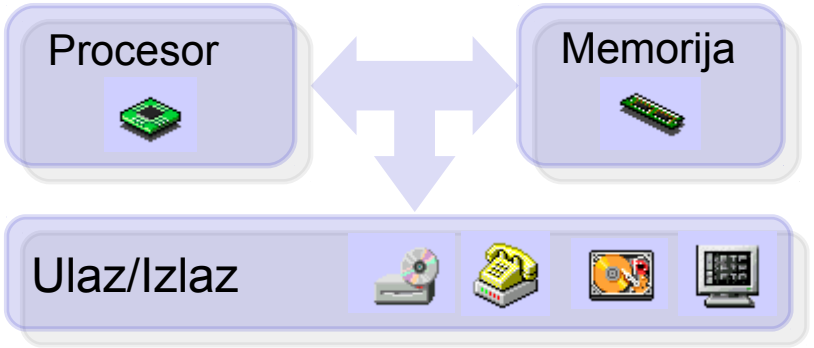
- **SW**

Dohvati instrukciju:	Instruction ← MEM[PC]
Pročitaj registre:	base ← Reg(Rs), data ← Reg(Rt)
Sračunaj adresu:	address ← base + sign_extend(imm16)
Zapiši u memoriju:	MEM[address] ← data
	PC ← PC + 4

- **J**

Fetch instruction:	Instruction ← MEM[PC]
Target PC address:	target ← PC[31:28] || Imm26 || ‘00’
Jump:	PC ← target

Komponente računara



- CPU aktivna komponenta računara koja odrađuje sav posao procesiranja podataka i donošenja odluka
- Sastoji se od:
  - Staza podataka (datapath) je dio procesora u kojem se odrađuju sve operacije nad podacima
  - Kontrola (control) je dio procesora u kojem se donose odluke koju operaciju u datom trenutku treba da izvrši datapath

CPU

- CPU se može posmatrati kao hardverski implementiran interpreter mašinskih instrukcija ISA specifikacije
  - Izvršavanje instrukcija unutar jednog hardverskog bloka previše kompleksno i neefikasno
- Datapath se zbog toga dijeli u komponente koje su međusobno povezane i odrađuju pojedine faze u izvršavanju instrukcije.
  - jednostavnije za dizajniranje i optimiziranje

Datapath faze

- MIPS instrukcije tokom izvršavanja odrađuju slične generalne korake koji se mogu podjeliti u pet faza:
- Faza 1: preuzimanje instrukcije (instruction fetch)
  - Bez obzira na tip instrukcije, 4 byte-na riječ koja opisuje slijedeću instrukciju mora se prvo pročitati iz memorije
  - U ovoj fazi inkrementira se i brojač tj PC=PC+4

Datapath faze

- Faza 2: Dekodiranje instrukcije (instruction decode)
  - Iz preuzete instrukcije dekodiraju se polja
    - Prvenstveno opcode na osnovu kojeg se dalje detektuju dužine i sadržaj ostalih polja
  - U ovoj fazi vrši se i čitanje registara u skladu sa dekodiranim poljima
    - Dva registra za R tip instrukcije
    - Jedan registar za I tip instrukcije
    - Bez čitanja za J tip instrukcije

Datapath faze

- Faza 3: ALU (Arithmetic Logic Unit)
  - U ovoj komponenti se obavlja većina instrukcija: aritmetičke (+, *, /, -), logičke (&,|) poređenje (slt) itd
  - Instrukcije load i store koriste ovu komponentu, npr za operaciju:
    - lw $s0, 16($t0)
    - potrebno je izvršiti sabiranje broja 16 sa adresom koja se nalazi u registru $t0
  - J instrukcija ne koristi ALU

Datapath faze

- Faza 4: Pristup memoriji (Memory access)
  - Većina instrukcija ne koristi ovu fazu te je ili preskaču ili ostaju neaktivne u ovoj fazi
  - Instrukcije load i store su jedine aktivne u ovoj fazi
- Faza 5: Pisanje u registre (Register write)
  - Većina instrukcija u ovoj fazi zapisuju rezultat izvršenja operacije u destinacijski registar
  - Instrukcije store, branch, jump koje ne zapisuju rezultat u registar su ili neaktivne ili preskaču ovaj korak

Kombinatorna i sekvencijalna kola

Kombinatorna kola

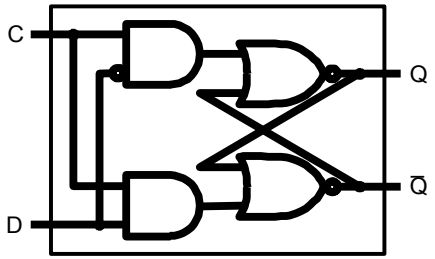
- Izlaz u potpunosti ovisi od ulaza
- Isti ulaz uvijek proizvodi isti izlaz.

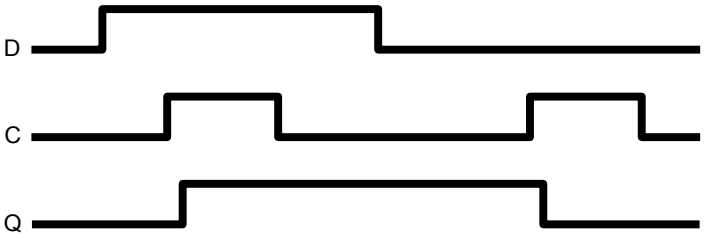
Sekvencijalna kola

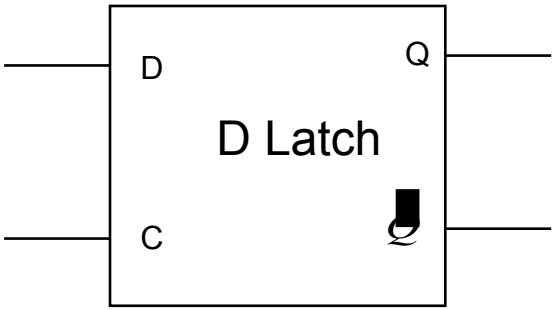
- Izlaz ovisi od kombinacije ulaza i stanja
- Isti ulaz može prouzrokovati različite izlaze
- Stanje se može I mijenjati na osnovu ulaza

D-latch

- Dva ulaza
  - Podatak koji treba pohraniti (D)
  - Clock signal (C) na osnovu kojeg se određuje trenutak promjene stanja
- Dva izlaza:
  - Vrijednost internog stanja Q i njegov komplement Q
- Kada je C = 1, D-latch kolo postavlja stanje Q na trenutnu vrijednost D
- Kada je C = 0, D-latch kolo zanemaruje D i zadržava trenutno Q

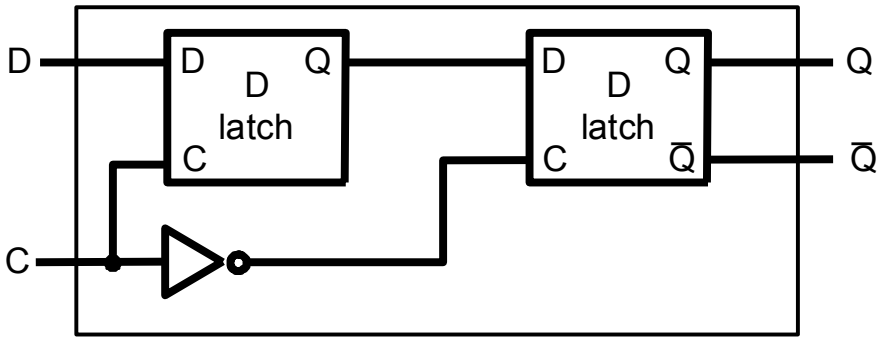






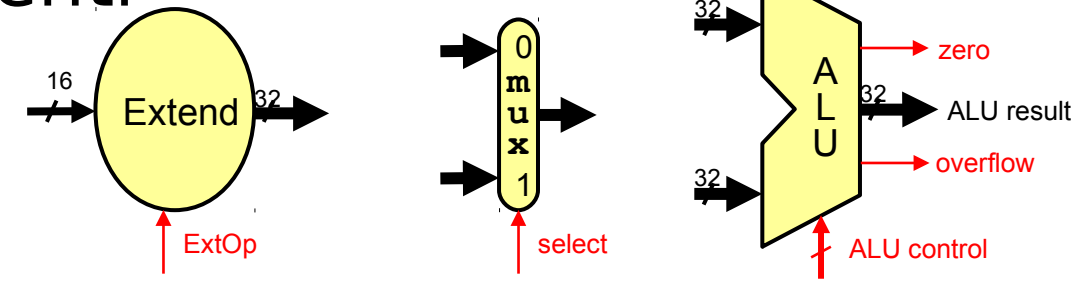
D flip-flop

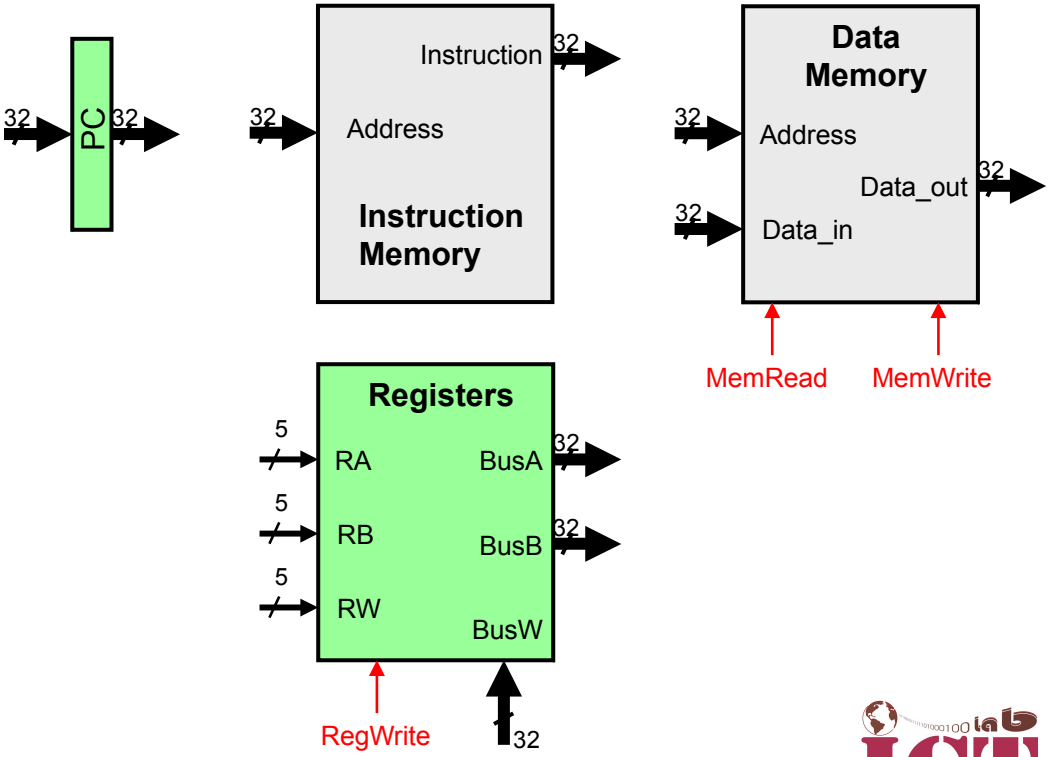
- Dva ulaza:
  - Podatak koji se pohranjuje (D)
  - Clock signal (C) na osnovu kojeg se određuje trenutak promjene stanja
- Dva izlaza:
  - Vrijednost internog stanja Q i njegov komplement
  - Interno stanje mijenja se na opadajuću ivicu clock signala



Datapath komponente

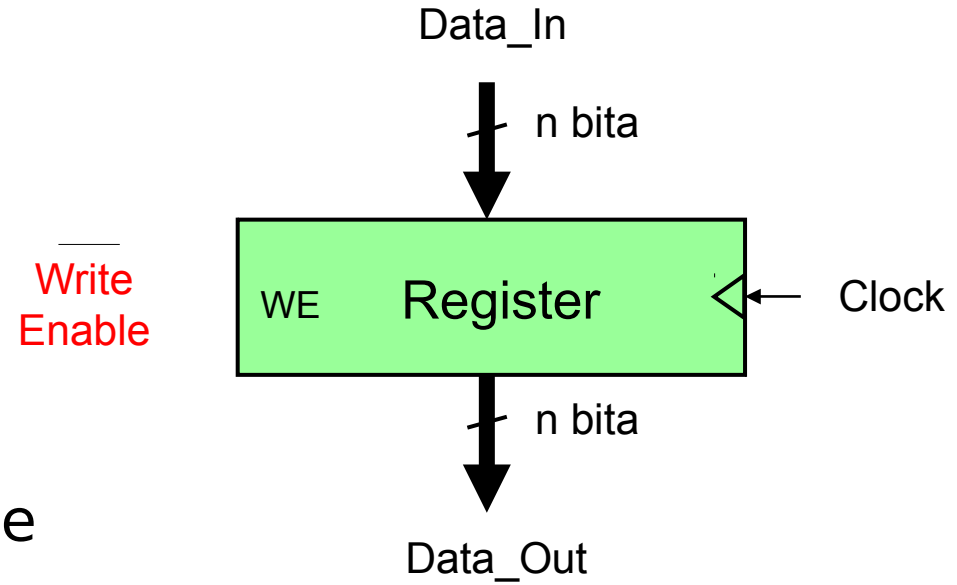
- Kombinatroni elementi
  - ALU, Sabirači
  - Ekstenderi
  - Multiplekseri
- Elementi sa stanjima
  - Memorija za instrukcije
  - Memorija za podatke
  - Registar fajl
  - PC registar





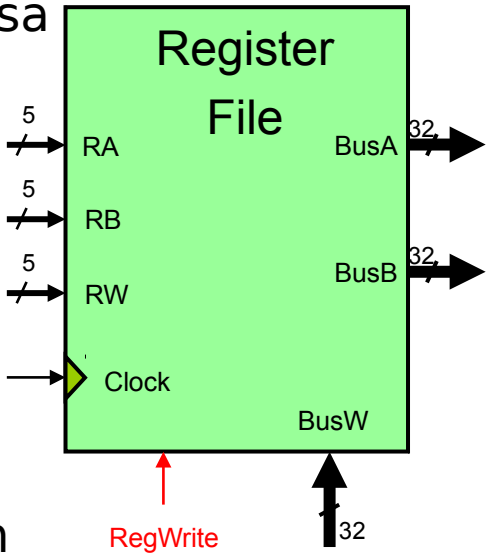
Registar

- Sličan D flip-flopu
  - n D flip flopova u paraleli
  - Povezani na isti clock
  - *Write Enable* (WE):
    - Ulaz dozvoljava pisanje
    - Kada je 0: *Data_Out* se ne mijenja
    - Kada je 1: *Data_Out* postaje *Data_In* nakon ivice *Clock* signala

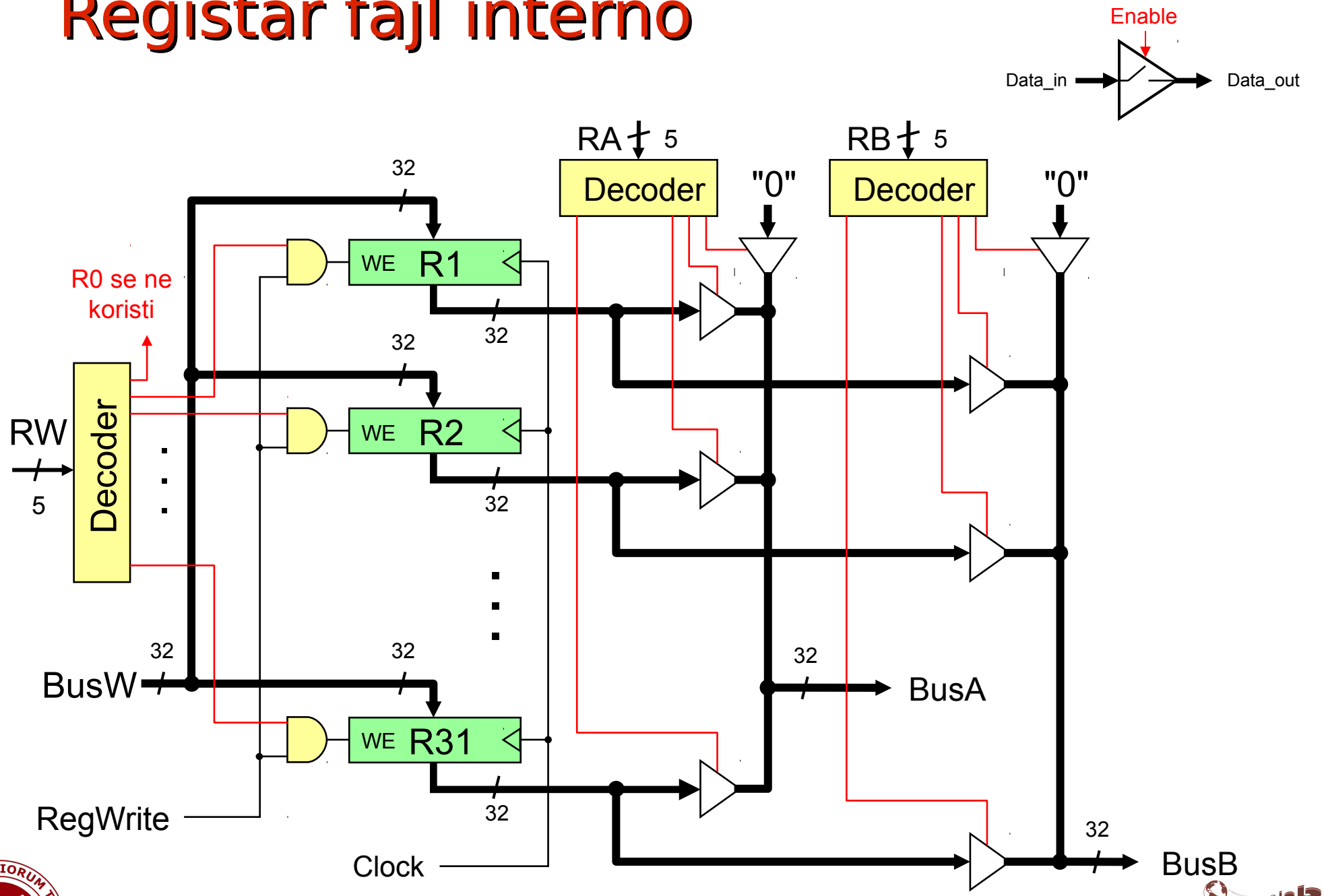


MIPS registar fajl

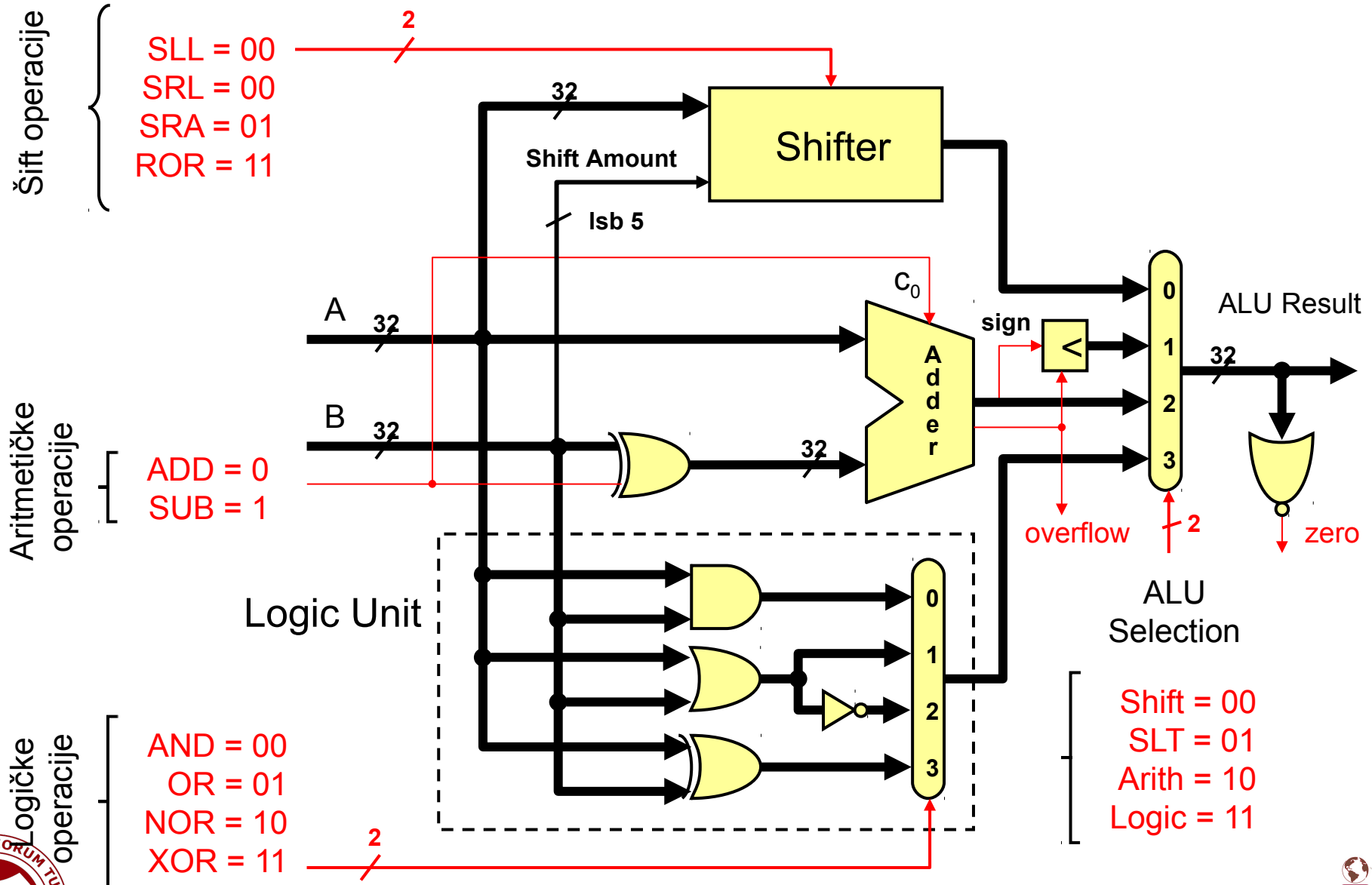
- Registar fajl sadrži trideset dva 32-bita registera
  - BusA and BusB: izlazi za čitanje stanja dva registra
  - BusW: podatak za upisivanje u registar fajl
    - Dva registra se čitaju a u jedan piše tokom jednog ciklusa
- Registri se biraju na osnovu:
  - RA selektira registar čije se stanje pojavljuje na BusA
  - RB selektira registar čije se stanje pojavljuje na BusB
  - RW selektira registar u koji se vrši upis podatka BusW
- Clock
  - Upisivanje podataka sinhronizirano je sa Clock signalom
  - Čitanje je asinhrona operacija, tj tokom čitanja registar fajl se može posmatrati kao kombinatorno kolo
    - Nakon što RA i RB postanu validni i stabilni BusA or BusB postaje validan nakon vremena koje se označava kao access time



Registar fajl interno

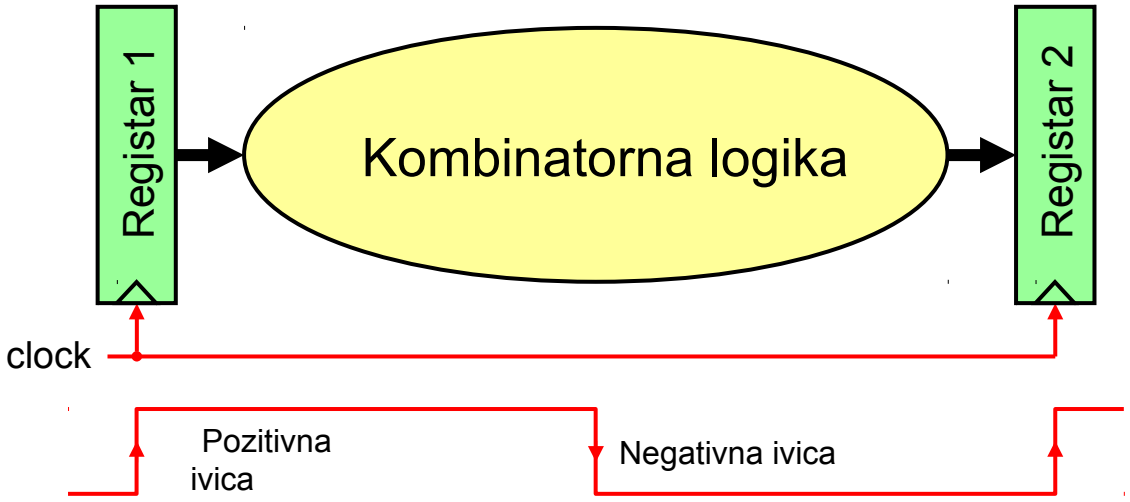


Primjer ALU implementacije



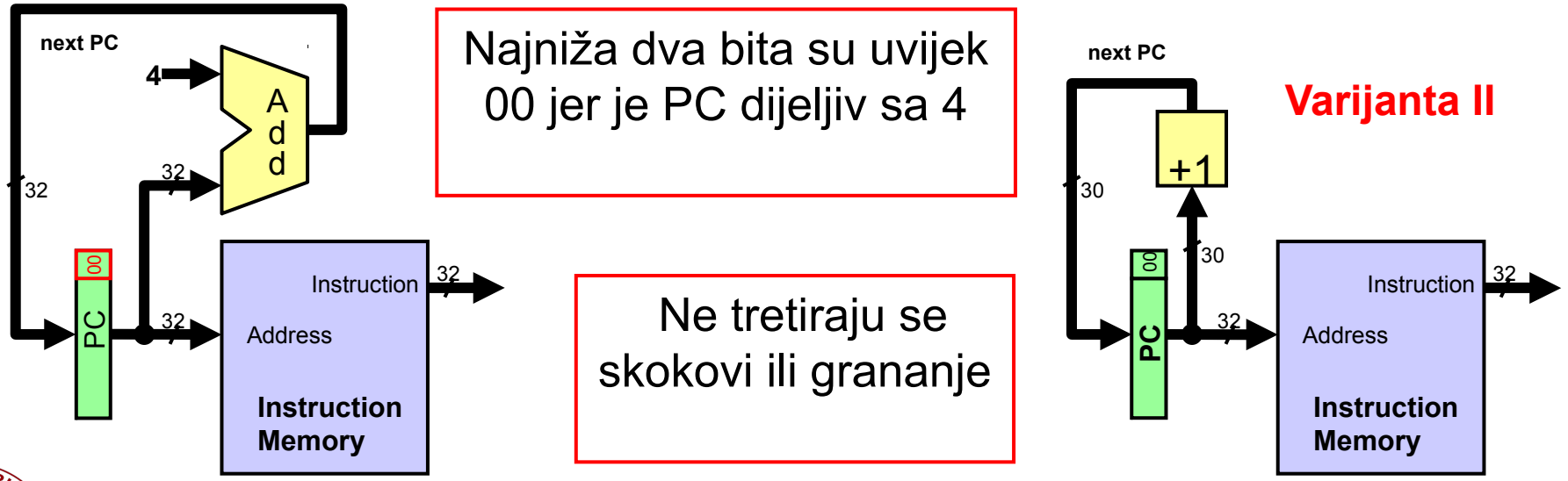
Jednociklusne operacije

- Clock signal određuje trenutke kada se vrijednost stanja može mijenjati
- Predpostavljamo da se promjene stanja svih elemenata vrše na istoj ivici clock signala
- Podaci moraju biti validni i stabilni prije pisanja
- Kod jednociklusne implementacije CPU-a sve datapath faze odvijaju se tokom jednog ciklusa clock signala

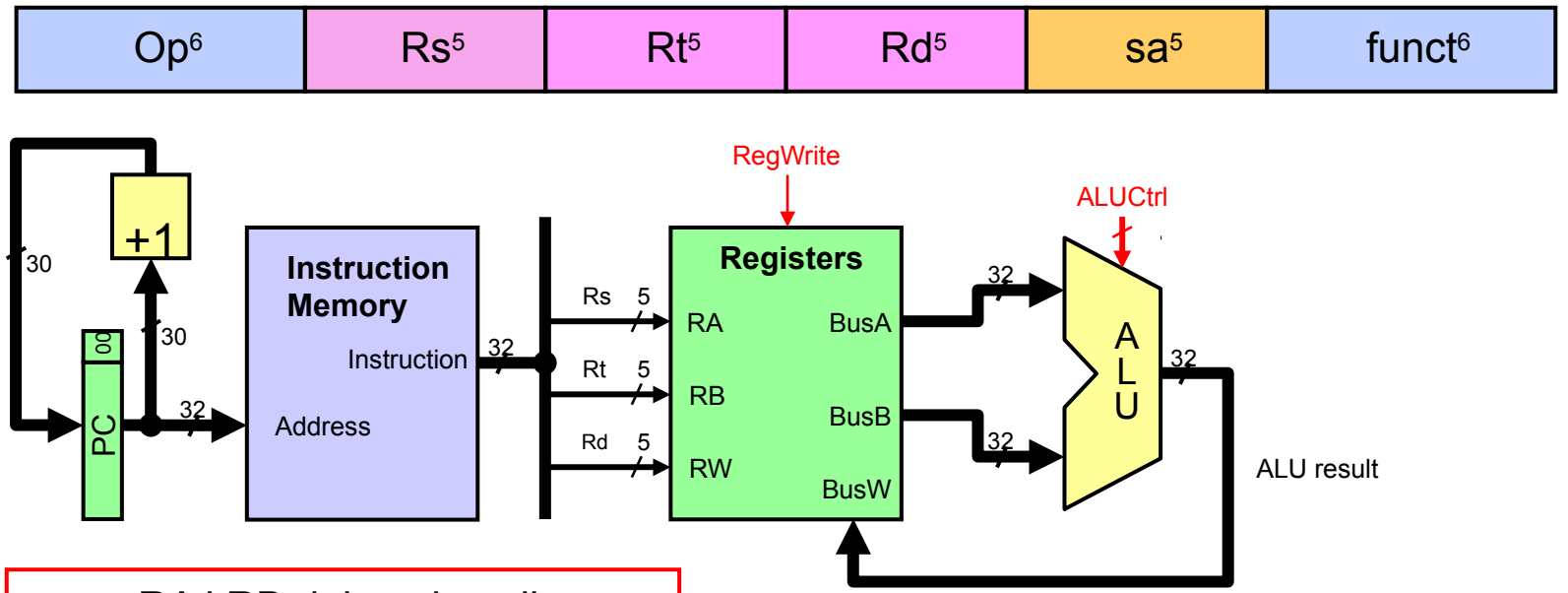


Datapath za čitanje instrukcija

- Potrebne komponente
  - PC registar
  - Memorija za instrukcije
  - Sabirač za povećanje PC-a



Datapath za R tip instrukcija



RA i RB dolaze iz polja instrukcije Rs i Rt

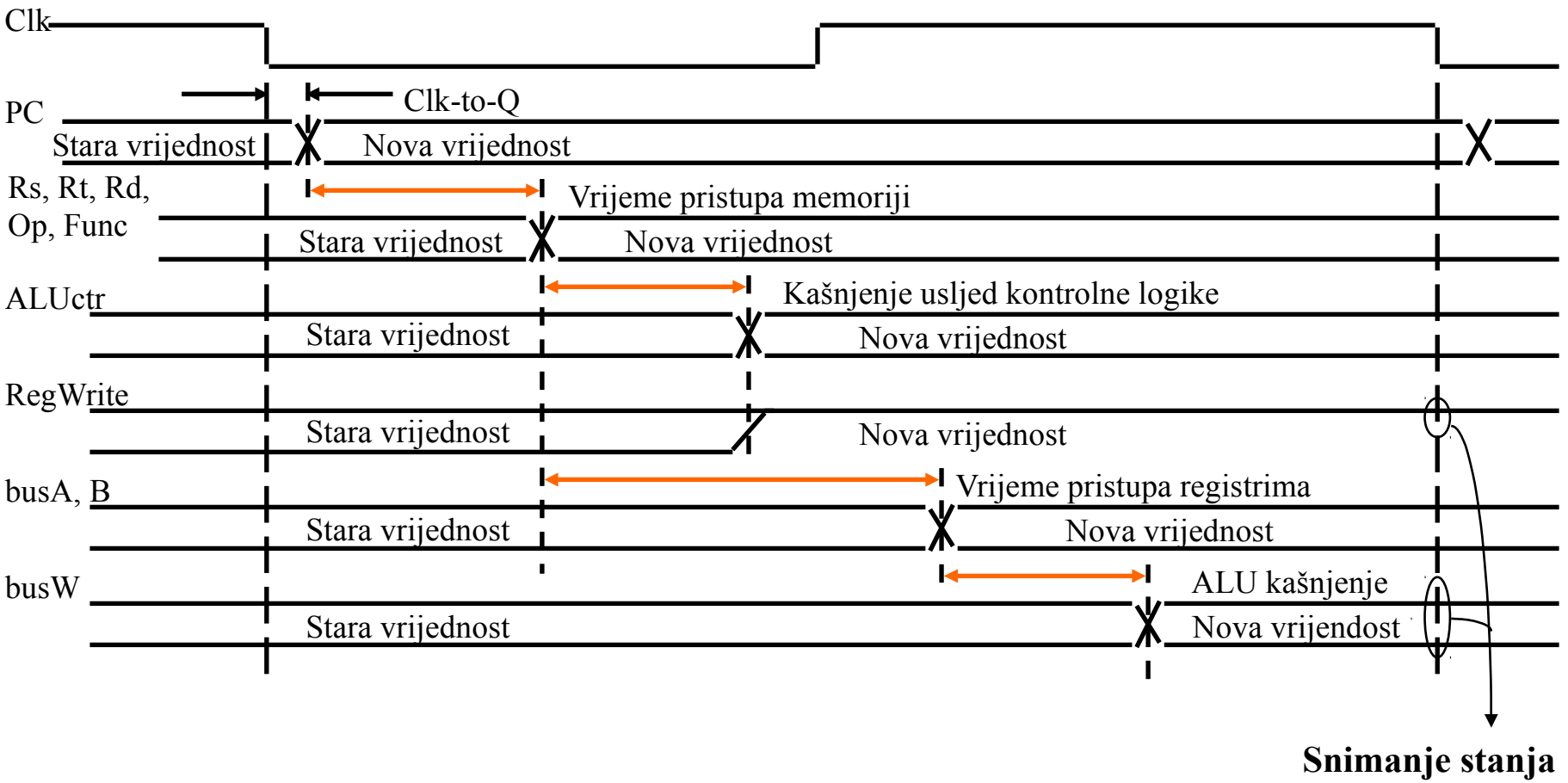
ALU ulazi dolaze od BusA i BusB

RW dolazi iz polja Rd

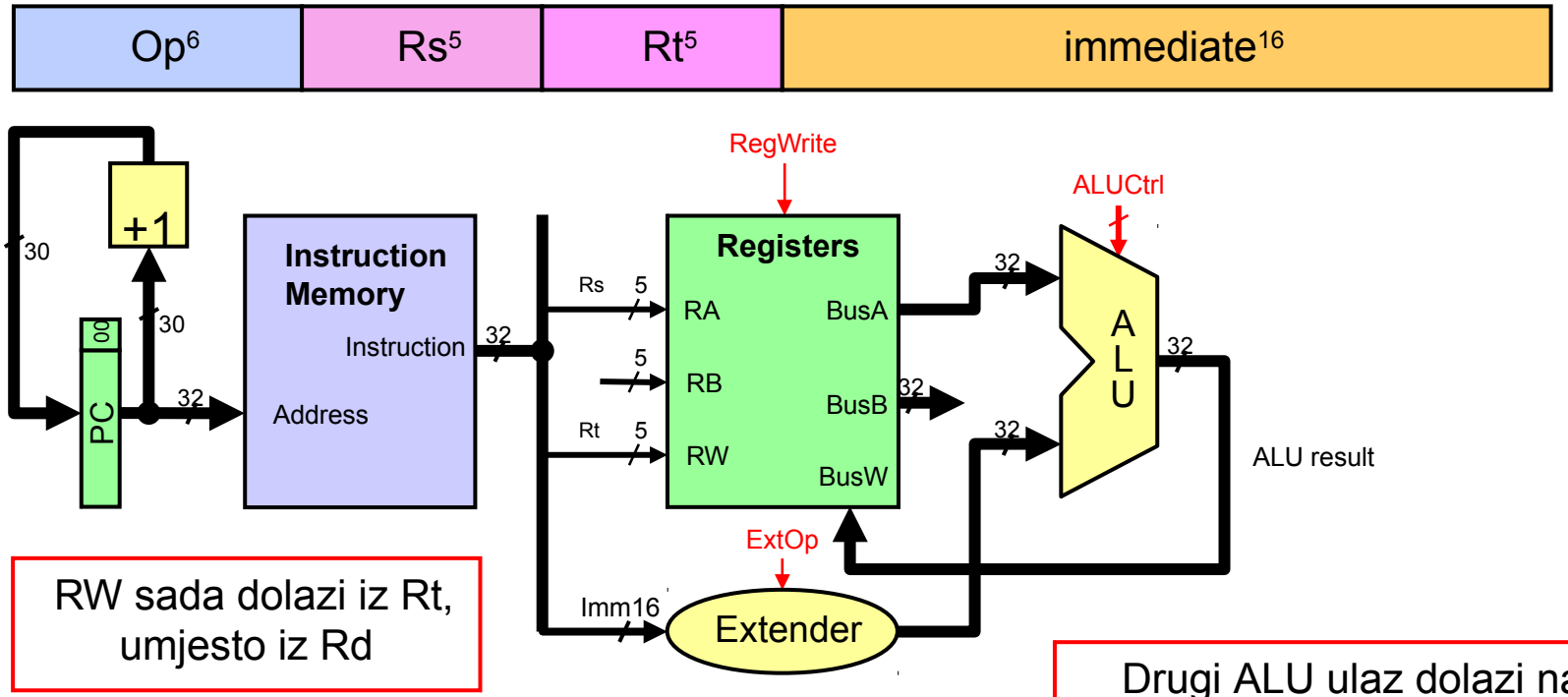
ALU izlaz se veže na BusW

- Kontrolni signali
  - ALUCtrl nastaje na osnovu funct polja instrukcije
  - RegWrite se koristi da se omogući pisanje u registar fajl

Vremenska analiza R datapath-a



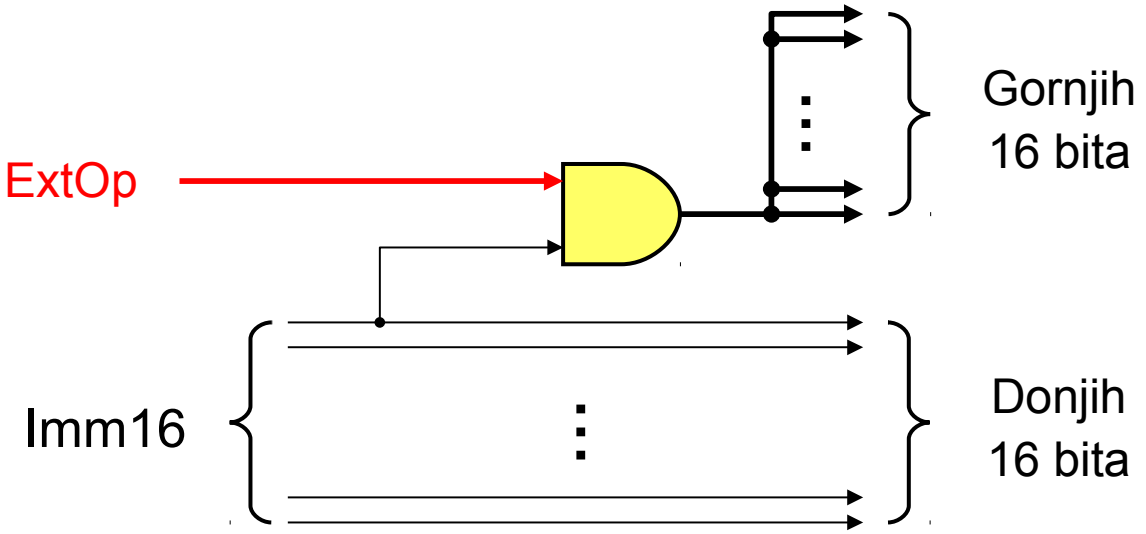
Datapath za I tip instrukcije



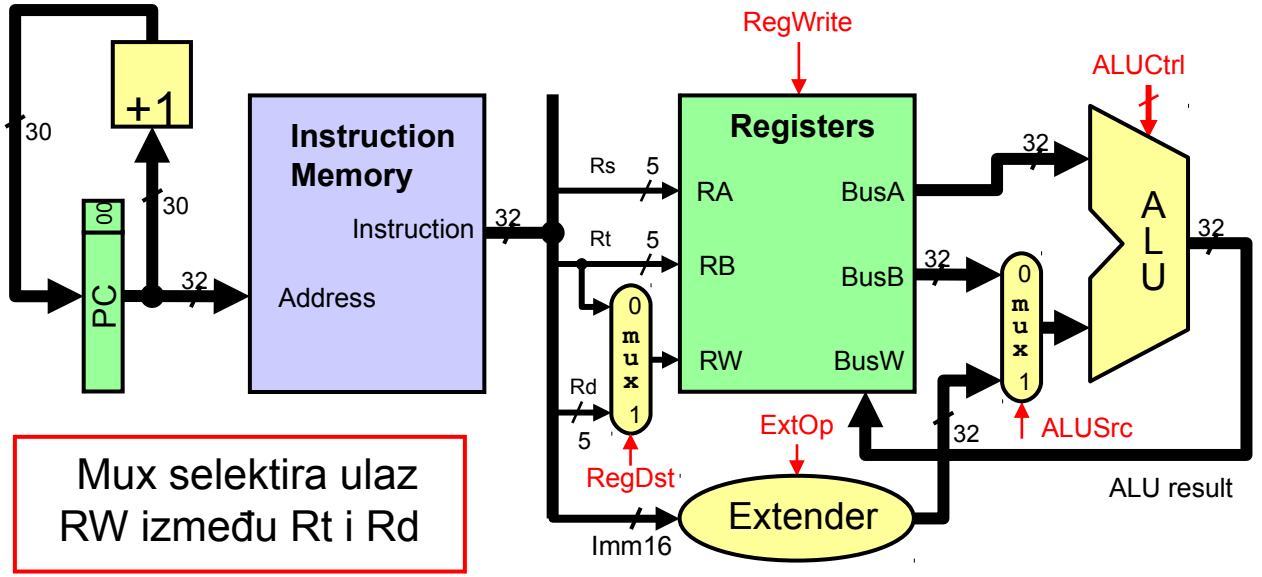
- Kontrolni signali
  - ALUCtrl nastaje na osnovu Op polja
  - RegWrite se koristi za omogućavanje snimanja u registar ALU result
  - ExtOp se koristi za kontrolu proširivanja 16-bitog immediate polja

Ekstender

- Dva tipa proširivanja
  - Proširivanje nulom
  - Proširivanje na osnovu znaka
- Kontrolni signal ExtOp indicira tip ekstenzije



Kombinovani Datapath

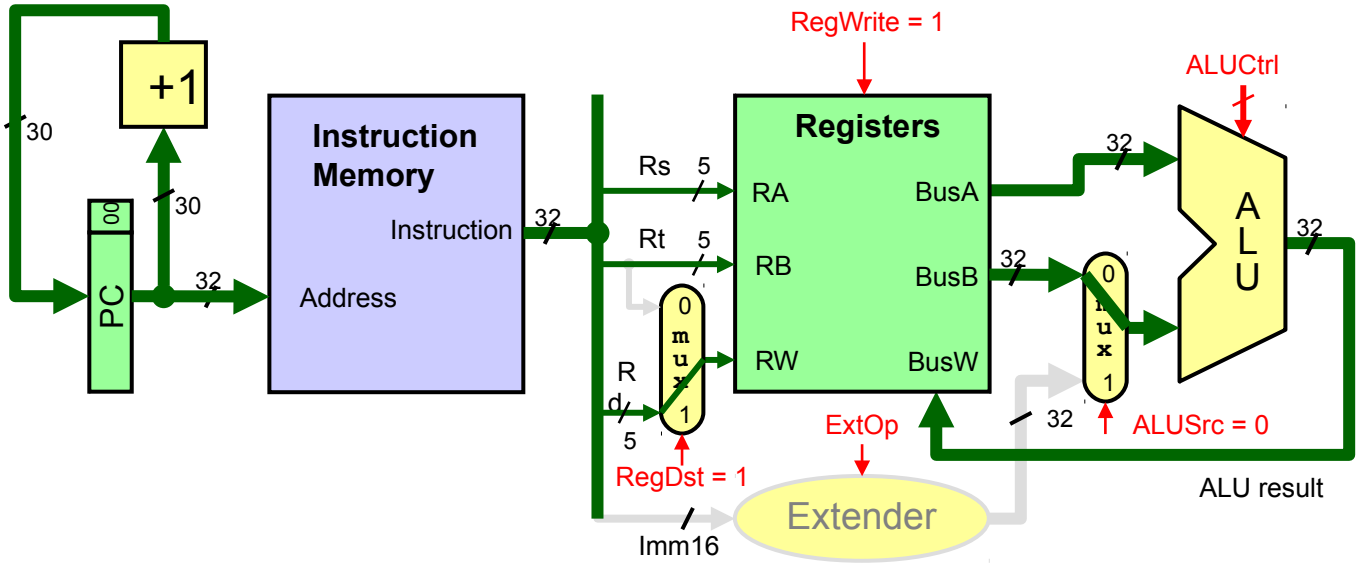


Mux selektira ulaz RW između Rt i Rd

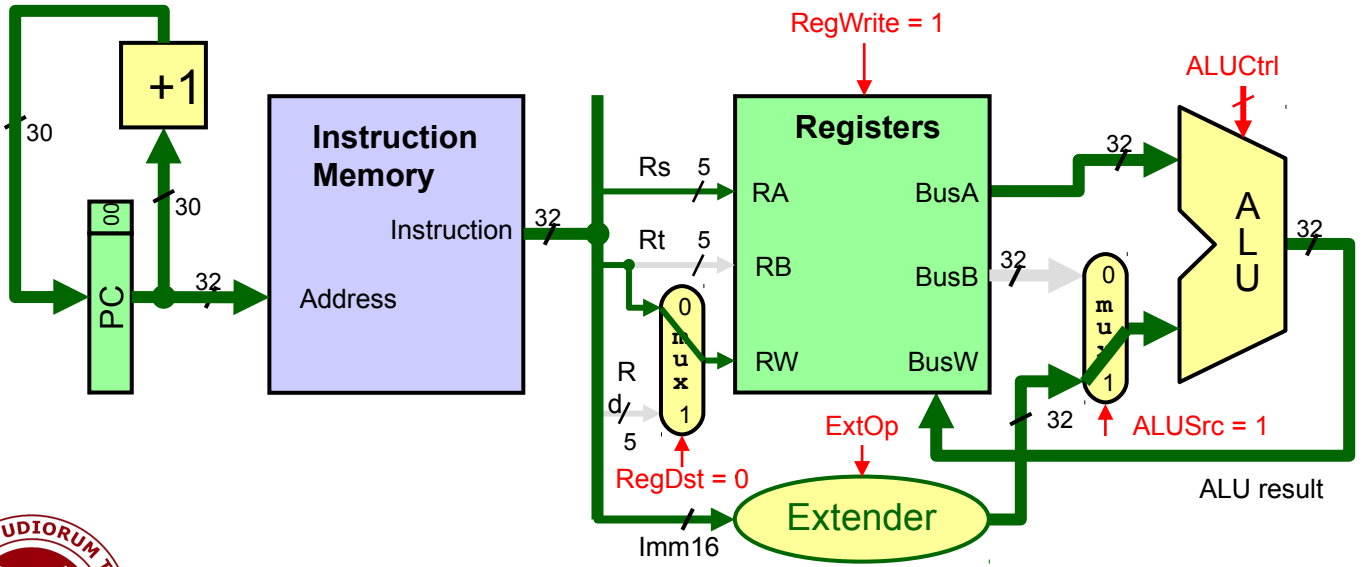
Drugi mux selektira drugi ALU ulaz između BusB izlaza registar fajla i konstante Imm16 iz instrukcije

- Kontrolni signali
  - ALUCtrl nastaje na osnovu polja Op ili funct
  - RegWrite omogućava zapisivanje ALU result
  - ExtOp kontrolira proširivanje 16-bitne konstante
  - RegDst 0 za Rt, 1 za Rd
  - ALUSrc 0 za BusB, 1 za Imm16

Kontroliranje kombinovanog Datapath-a



R-tip instrukcije. Aktivni dio datapath-a je prikazan **zelenom** bojom



I-tip instrukcije. Aktivni dio datapath-a je prikazan **zelenom** bojom