Arhitektura računara

dr.sc. Amer Hasanović

Pregled

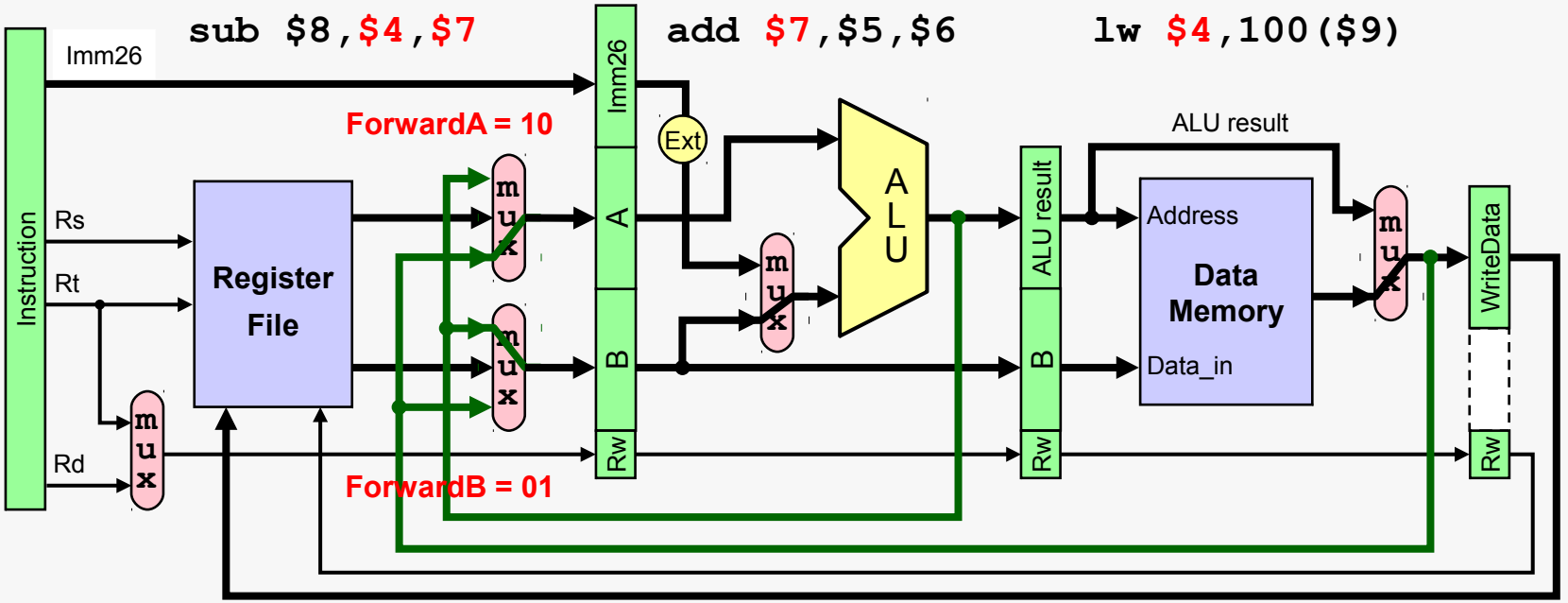
- Pipeline hazardi nastavak
  - RAW hazard nakon load operacije
  - Kontrolni hazard

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

Primjer RAW hazard

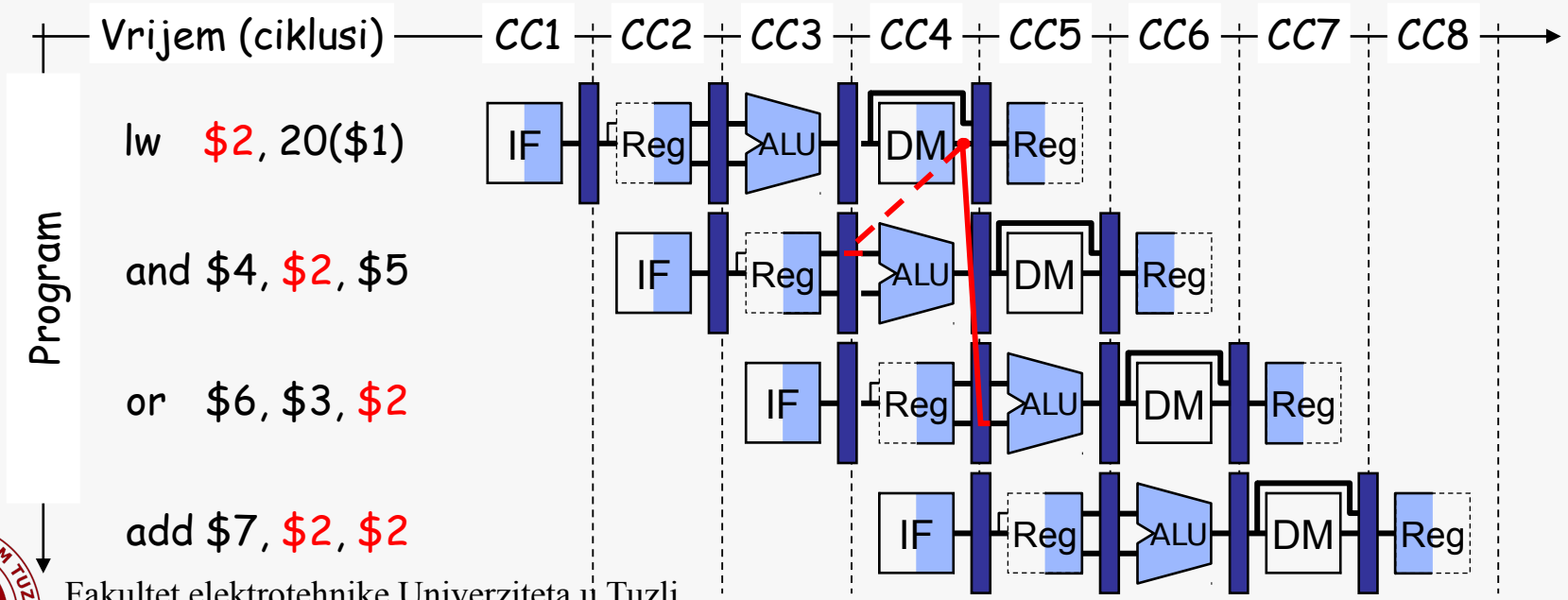
Instrukcije:

```
lw      $4, 100($9)
add     $7, $5, $6
sub     $8, $4, $7
```



Kašnjenje usljed load instrukcije

- Load instrukcija ima kašnjenje koje se ne može riješiti proslijeđivanjem
- U primjeru
  - lw instrukcija nema podatak sve do kraja CC4
  - and instrukcija zahtijeva podatak na početku CC4
  - proslijeđivanje je moguće odraditi tek za instrukciju or



Detektovanje RAW hazarda nakon load

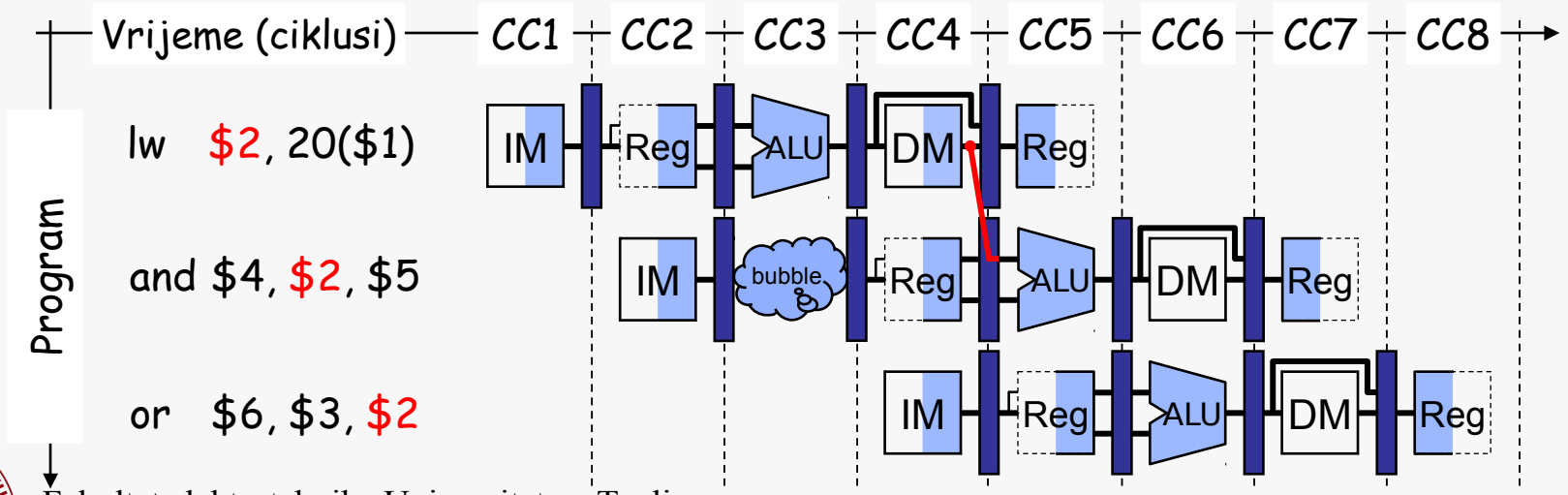
- Za detektovanje hazarda potrebni su uslovi:
  - load instrukcija je u ID/EX registru
  - Instrukcija koja treba podatak pročitan od load instrukcije je u IF/ID register
- tj:

  if ((ID/EX.MemRead == 1) and (ID/EX.Rw ≠ 0) and

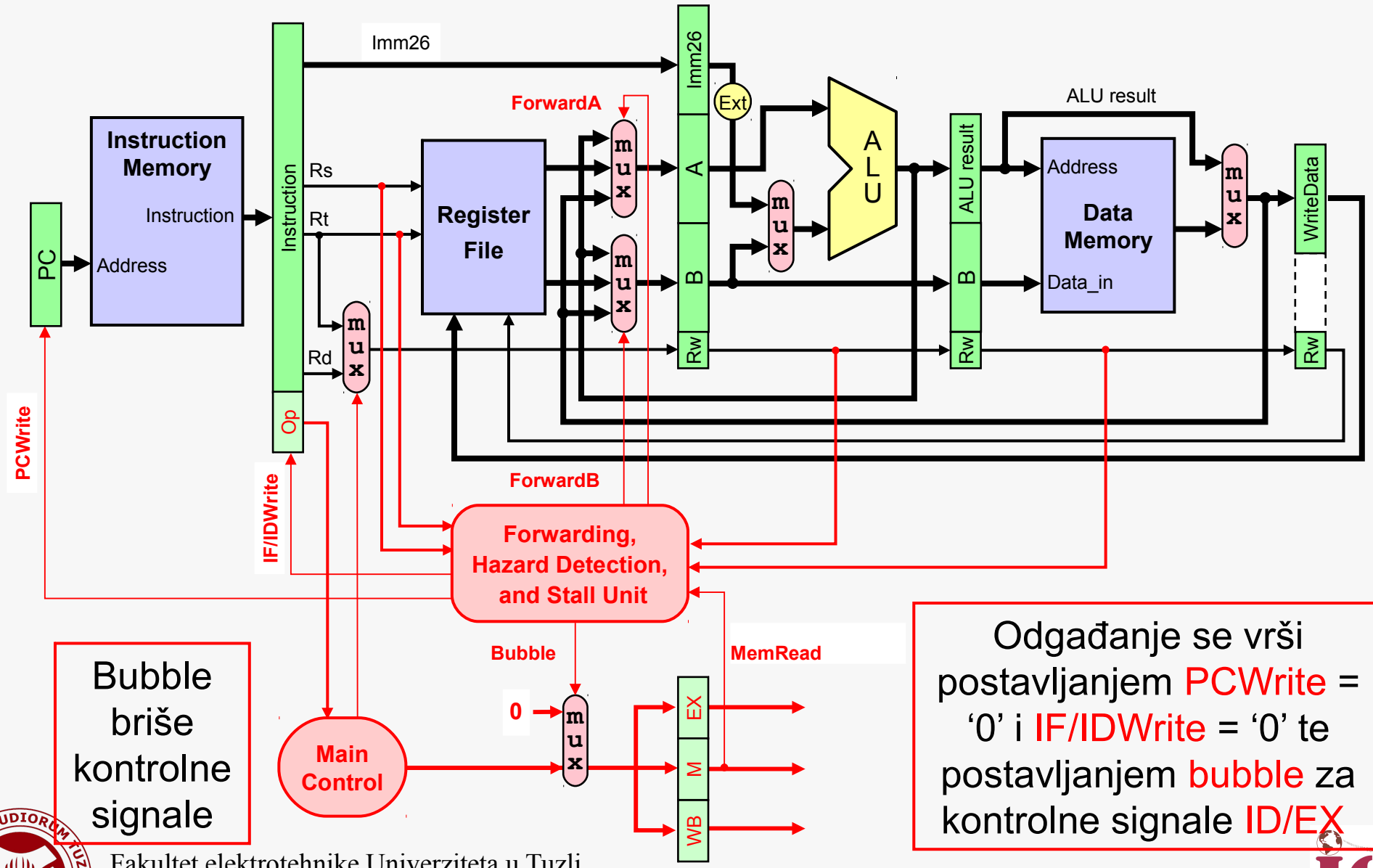
  ((ID/EX.Rw == IF/ID.Rs) or (ID/EX.Rw == IF/ID.Rt))) Stall
- Rješava se ubacivanjem bubble (mjehurića) nakon load instrukcije
  - Mjehurić je suštinski nop operacija koja troši jedan ciklus

Pipeline interlock

- Pipeline interlock hardver za detekciju i tretman load hazarda
  - Originalni MIPS je bez pipeline interlock hardvera
  - Zadržava stare vrijednosti PC i IF/ID registara
    - ne dohvata se nova instrukcija, a instrukcija nakon load odgođena je za jedan ciklus
- Implementira se unošenjem mjehurića (bubble) u ID/EX registre



Pipeline interlock implementacija



Bubble briše kontrolne signale

Odgađanje se vrši postavljanjem PCWrite = '0' i IF/IDWrite = '0' te postavljanjem bubble za kontrolne signale ID/EX

Kompajler optimiziranje koda

- Kompajleri tokom prevođenja koda mogu izvršiti preinake u redosljedu izvođenja kako bi se izbjegla odgađanja instrukcija u cjevovodu

- Neka je dat C kod cegment:

```
a = b + c;  d = e – f;
```

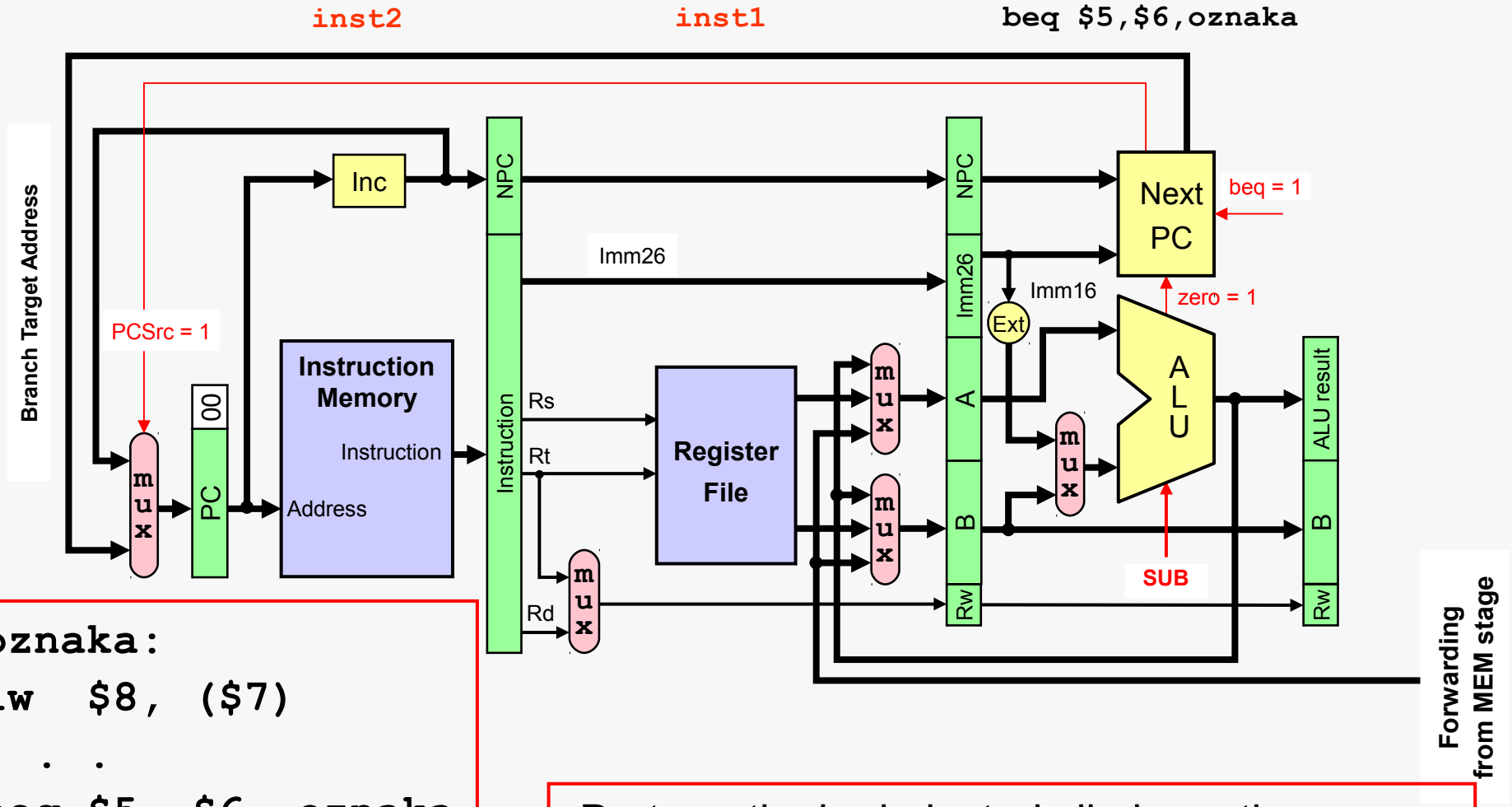
Neoptimiziran kod

```
lw   $10, ($1)       # $1 = adr b
lw   $11, ($2)       # $2 = adr c
add  $12, $10, $11   # stall
sw   $12, ($3)       # $3 = adr a
lw   $13, ($4)       # $4 = adr e
lw   $14, ($5)       # $5 = adr f
sub  $15, $13, $14   # stall
sw   $15, ($6)       # $6 = adr d
```

Optimiziran kod

```
lw   $10, 0($1)
lw   $11, 0($2)
lw   $13, 0($4)
lw   $14, 0($5)
add  $12, $10, $11
sw   $12, 0($3)
sub  $15, $13, $14
sw   $14, 0($6)
```

Kontrolni hazard: branch instrukcija

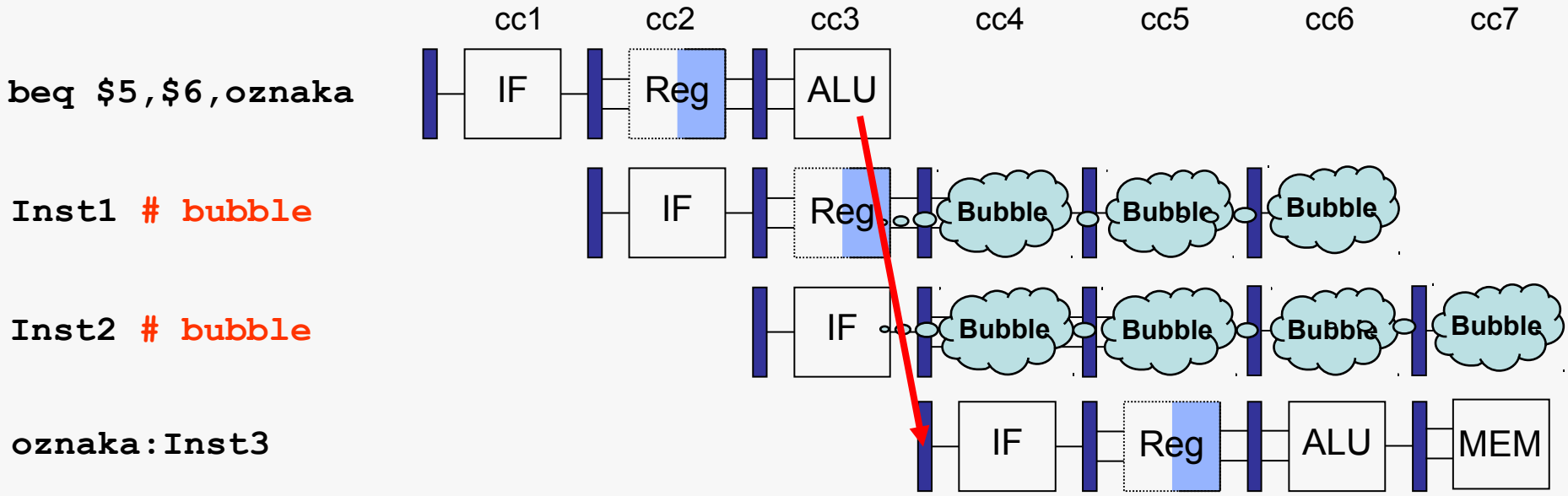



```
oznaka:
lw   $8, ($7)
. . .
beq $5, $6, oznaka
inst1
inst2
```

Do trenutka kada instrukcija beq stigne u ALU fazu, inst1 je u fazi dekodiranja a inst2 je u fazi preuzimanja.

Dvociklusno kašnjenje usljed branch

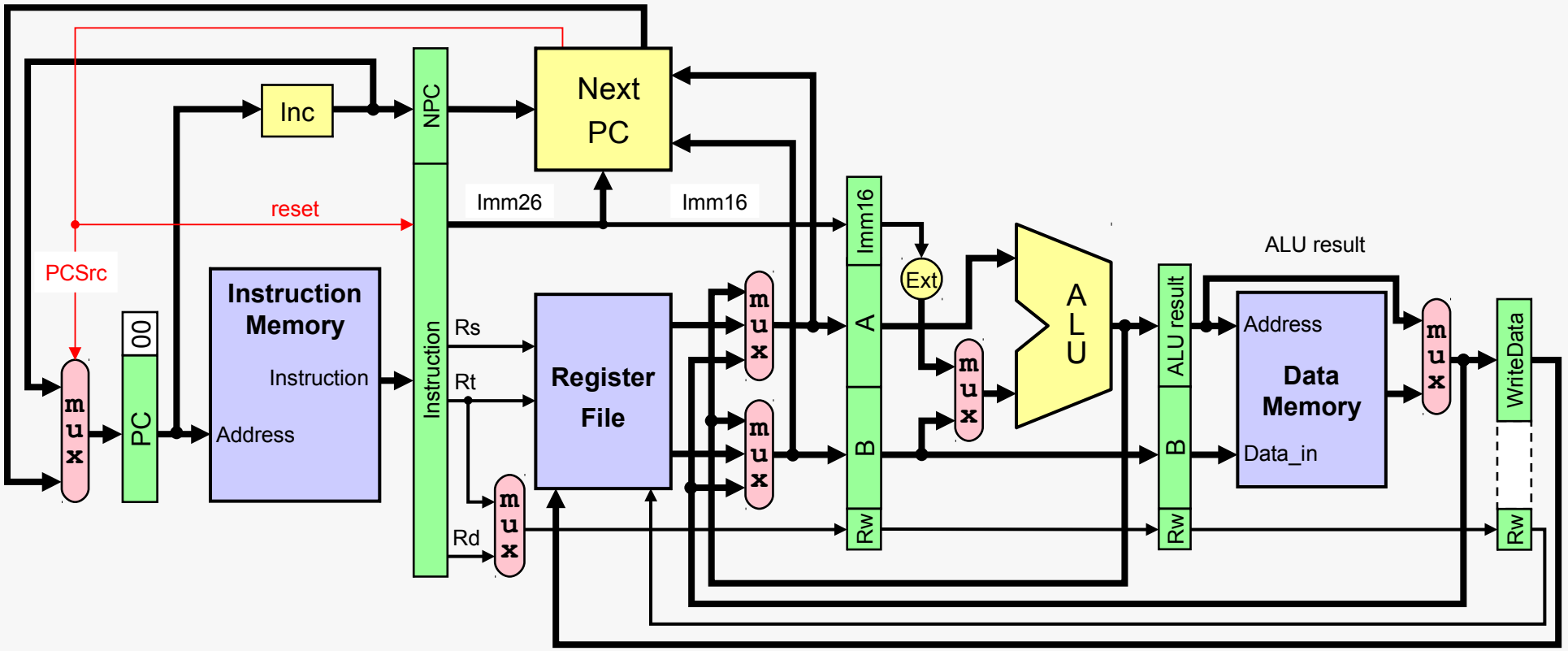
- Inst1 i Inst2 će biti dohvaćene
  - Rezultat tih instrukcija treba biti odbačen u slučaju da je potrebno izvršiti grananje
  - U suprotnom, instrukcije se trebaju izvršiti normalno



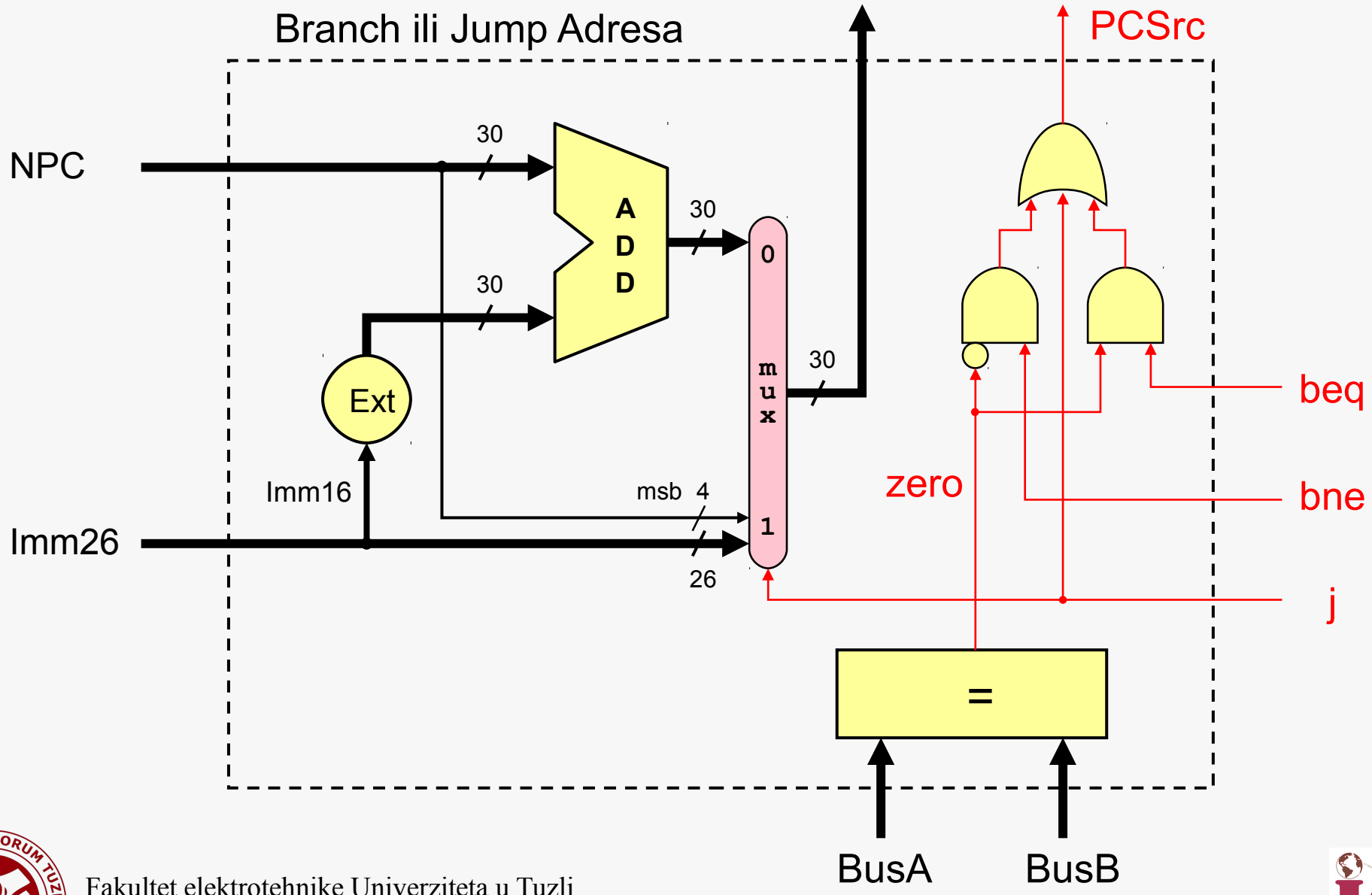
Reduciranje kašnjenja usljed branch

- Kašnjenje usljed branch instrukcije moguće je svesti na jedan ciklus
- Grananje je moguće odrediti već u fazi dekodiranja
  - Next PC blok se pomjera u ID fazu
    - Komparator se dodaje u Next PC logiku
- Sada se preuzima samo jedna instrukcija prije odluke o grananju.
- Tretman hazarda:
  - Varijanta 1: U slučaju da se izvršava grananje preuzeta instrukcija konvertuje se u nop resetovanjem IF/ID registra (predict not taken)
  - Varijanta 2: Preuzeta instrukcija se uvijek izvršava bez obzira na rezultat uslova grananja (delayed branch)

Datapath modifikacije



Next PC blok

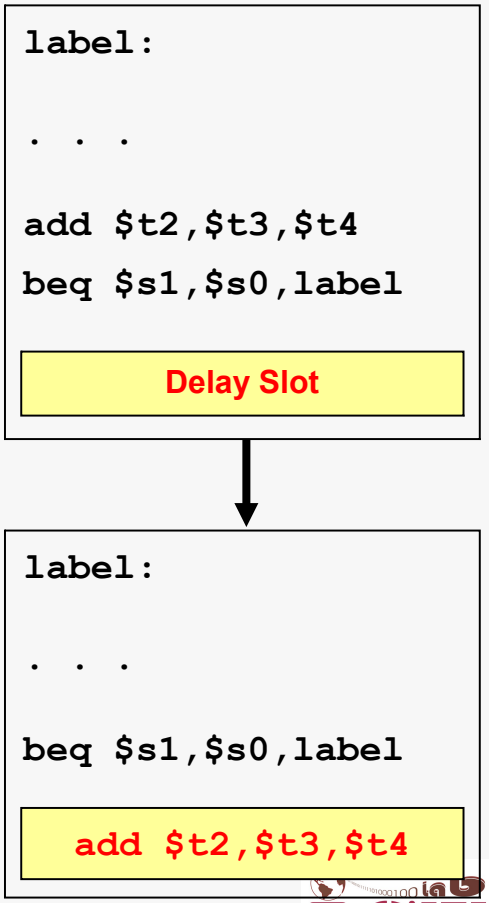

Branch ili Jump Adresa
PCSrc
NPC
30
30
30
A
D
D
0
m
u
x
1
30
Ext
Imm16
msb 4
Imm26
26
beq
zero
bne
j
=
BusA
BusB

Zakašnjela branch instrukcija (delayed branch)

- Promjeniti definiciju instrukcije branch na način da se eventualno grananje obavlja tek nakon slijedeće instrukcije
- Nakon branch instrukcije dodaje se jedan slot za kašnjenje
  delay slot

  branch instrukcija

  branch delay slot (slijedeća instrukcija)
- Kompajler obično popunjava slot
  - selektiranjem nezavisne funkcije
  - od prije grananja
- U slučaju da ne pronađe nezavisnu instrukciju
  - kompajler popunjava slot sa nop instrukcijom



Primjer

Bez delay slota

```
or   $8, $9 ,$10
add $1 ,$2,$3
sub $4, $5,$6
beq $1, $4, Exit
xor $10, $1,$11
Exit:
```

Sa delay slotom

```
add $1 ,$2,$3
sub $4, $5,$6
beq $1, $4, Exit
or   $8, $9 ,$10
xor $10, $1,$11
Exit:
```