Arhitektura i organizacija računara 2

Hazardi

Katedra za računarstvo
Elektronski fakultet u Nišu
Univerzitet u Nišu

Hazardi

- **Instrukcioni hazardi (zavisnosti)** javljaju se u situacijama kada instrukcije čitaju ili upisuju u registre koji se koriste od strane drugih instrukcija
  - **RAR**
  - **RAW**
  - **WAR**
  - **WAW**
- **Upravljački (*control*) hazardi**
- **Strukturni hazardi**

RAR

RAR (*read after read*) hazardi javljaju se kada dve instrukcije čitaju isti sadržaj.

```
add   r1, r2, r3
sub   r4, r5, r3
```

Obe instrukcije čitaju sadržaj registra *r3* uzrokujući pojavu RAR hazarda.

- RAR hazardi ne uzrokuju problem procesoru jer se operacijom čitanja sadržaja registra ne menja njegov sadržaj, tj. operacija tipa *Read* nije destruktivna.

- Dve instrukcije izmedju kojih postoji RAR hazard mogu da se izvršavaju u sukcesivnim ciklusima (ili u isti ciklus, kod onih procesora koji izvršavaju više od jedne instrukcije/ciklusu).

RAW

RAW (*read after write*) hazardi – javljaju se kada instrukcija čita sadržaj registra u kome se vrši upis od strane prethodne instrukcije.

```
add r1, r2, r3
sub r4, r5, r1
```

Instrukcija ***sub*** čita izlaz koji generiše instrukcija ***add*** (tj. čita saržaj registra *r1*), uzrokujući pojavu hazard tipa RAW.

- RAW hazardi su poznati kao ***data dependencies*** ili ***true dependencies*** (***prava zavisnost***), zbog toga što se javljaju u trenutku kada tekuća instrukcija treba da koristi rezultat generisan od strane druge instrukcije.

- Kada se javi RAW hazard, instrukcija koja čita sadržaj nekog registra ne može da produži sa procesiranjem sve dok instrukcija koja vrši upis u taj registar ne prodje procesiranje kroz WB stepen.
  - tj. podatak koji je potreban instrukciji koja vrši čitanje još nije dostupan u tom trenutku.

WAR i WAW

- **WAR (*write-after-read*)** i **WAW (*write-after-write*)** hazardi se javljaju u situacijama kada se u izlazni registar instrukcije čita, ili se u izlazni registar instrukcije upisuje, od strane prethodne instrukcije.
- Ovi tipovi hazarda se ponekad nazivaju ***name hazardi***, jer se javljaju zbog razloga što procesor u svom registarskom polju poseduje konačan broj registara.
- U slučaju kada procesor poseduje beskonačan (neograničeni) broj registara, tada on može da koristi uvek različiti registar za čuvanje rezultata svake od instrukcija (po jedan za čuvanje rezultata svake od instrukcija), tako da do WAR i WAW hazarda potencijalno nikad neće doći (javiti se).

WAR i WAW

```
add   r1, r2, r3
sub   r2, r5, r6
```

Operacija *sub* upisuje u *r2*, čiji se sadržaj čita od strane instrukcije *add*, kreirajući hazard tipa WAR

```
add   r1, r2, r3
sub   r1, r5, r6
```

Operacija *sub* upisuje u isti registar kao i *add*, kreirajući hazard tipa WAW.

WAR i WAW

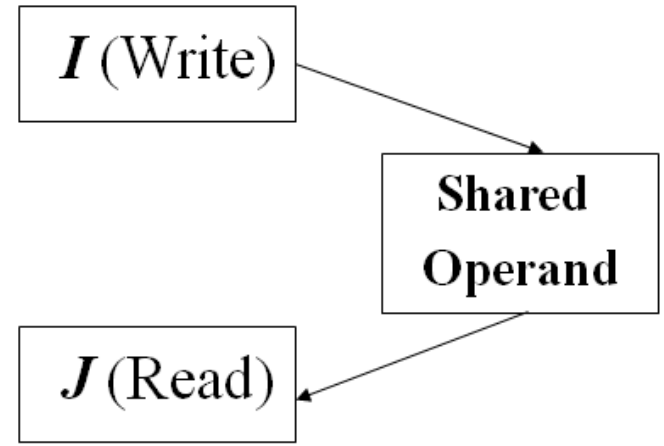
- **Ako procesor izvršava instrukcije po redosledu u kome se one pojavljuju u programu i koristi isti protočni sistem za sve instrukcije**, tada **WAR** i **WAW** hazardi ne uzrokuju efekat kašnjenja (mehurova).

- S obzirom da se u odredišni registar instrukcije vrši upis tek u stepenu WB, instrukcije izmedju kojih postoje WAW hazardi ulaze u stepen WB u redosledu kako se one pojavljuju u programu i upisuju rezultate u odredišne registre u pravilnom redosledu.

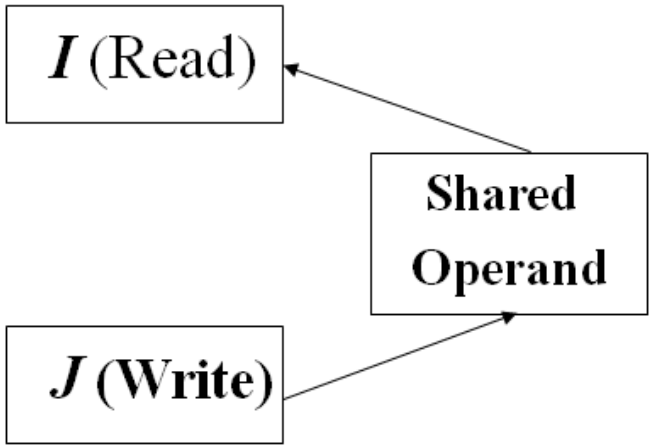
WAR i WAW

- Za slučaj da sve instrukcije procesora nemaju istu latenciju WAW i WAR hazardi mogu da uzrokuju probleme jer može da se desi da se instrukcija sa kraćom latencijom završi pre nego što završi instrukcija sa dužom latencijom.
  - Ovi procesori mora da vode računa o ***name dependencies*** koje se javljaju izmedju instrukcija, i pri tome zaustave protočni sistem, ako je neophodno, kako bi uspešno rešili hazarde.
  - WAR i WAW hazardi su tipični za procesore kojih karakteriše ***out-of-order-execution***.

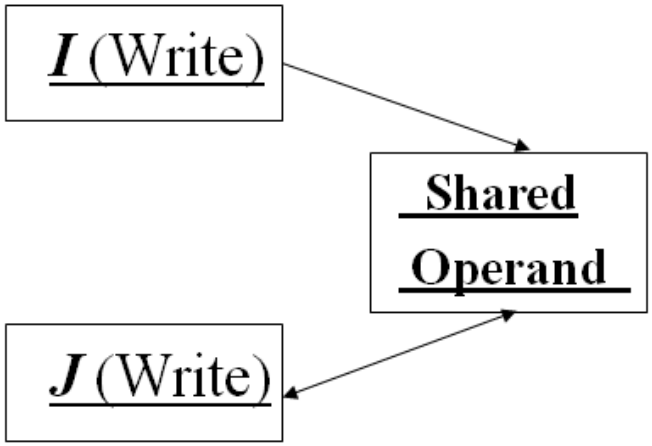
Instrukcioni hazardi



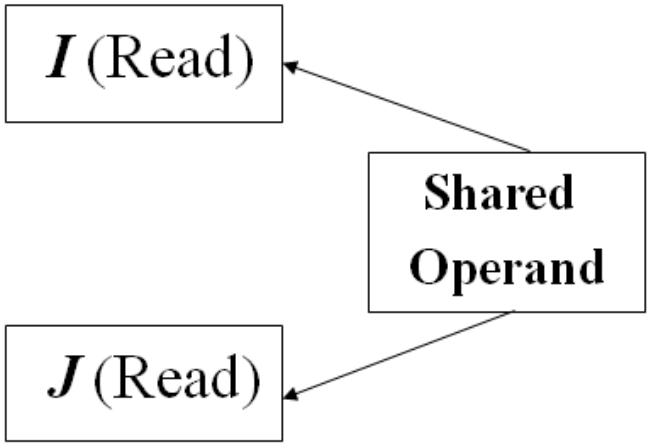
Read after Write (RAW)



Write after Read (WAR)



Write after Write (WAW)



Read after Read (RAR) not a hazard

Upravljački (*control*) hazardi

- **Instrukcije tipa *Branch*** mogu da uzrokuju zastoj u radu protočnog sistema iz razloga što procesor ne može da odredi koju instrukciju da pribavi kao narednu sve dok se ne izvrši instrukcija *Branch*.

- Kašnjenje koje postoji od trenutka kada instrukcija *Branch* udje u protočni sistem do trenutka kada naredna istrukcija u programu udje u protočni sistem naziva se ***branch-delay*** procesora.

- ***Branch-delay*** se često nazivaju ***control*-hazardi** iz razloga što je kašnjenje posledica programskog toka upravljanja.

- ***Branch-delay*** ima značajan uticaj na performanse savremenih procesora, pa je zbog toga razvijen veliki broj tehnika za njihovo uspešno rešavanje.

Strukturni hazardi

- Strukturni hazardi se javljaju kada hardver protočnog procesora nije u stanju da istovremeno izvrši sve instrukcije.
  - Tako na primer, ako RF polje nema dovoljno portova, koji će obezbediti instrukciji u stepenu WB da u toku istog ciklusa upiše rezultat u RF polje, kako bi one mogle da se čitaju od strane stepena RR tada je potrebno zaustaviti napredovanje instrukcije u stepenu RR.
- Strukturni hazardi kod jednostavnih protočnih procesora su retki, ali kod procesora koji izvršavaju više od jedne instrukcije po taktnom ciklusu unose ozbiljna ograničenja.

Zadatak 1

- Za datu kodnu sekvencu izlistati sve zavisnosti-po-podacima (*data dependencies*) tipa RAW, WAR i WAW.

```
I0:   A = B + C ;
I1:   C = A – B ;
I2:   D = A + C ;
I3:   A = B * C * D ;
I4:   C = F / D ;
I5:   F = A - G ;
I6:   G = F + D ;
```

Zadatak 1 - rešenje

I0: A = B + C ;
I1: C = A – B ;
I2: D = A + C ;
I3: A = B * C * D ;
I4: C = F / D ;
I5: F = A - G ;
I6: G = F + D ;

RAW zavisnost		**WAR** zavisnost		**WAW** zavisnost	
od instrukcije	ka instrukciji	od instrukcije	ka instrukciji	od instrukcije	ka instrukciji
I0	I1	I0	I1	I0	I3
I0	I2	I1	I3	I1	I4
I1	I2	I2	I4		
I3	I5	I3	I4		
I2	I3	I4	I5		
I1	I3	I5	I6		
I2	I4				
I3	I5				
I5	I6				

Zadatak 2

U sledećoj programskoj sekvenci identifikovati

- RAW hazarde,
- WAR hararde,
- WAW hazarde
- Upravljačke (*control*) hazarde

```
1. Div   r2,  r5,  r8
2. Sub   r9,  r2,  r7
3. Ash   r5,  r14, r6
4. Mul   r11, r9,  r5
5. Beq   r10, #0,  r12
6. Or    r8,  r15, r2
```

Zadatak 2 - rešenje

```
1. Div    r2,  r5,  r8
2. Sub    r9,  r2,  r7
3. Ash    r5,  r14, r6
4. Mul    r11, r9,  r5
5. Beq    r10, #0,  r12
6. Or     r8,  r15, r2
```

- RAW hazardi postoje izmedju sledećih instrukcija:

  1-2; 3-4; 2-4; 1-6

- WAR hazardi postoje izmedju sledećih instrukcija:

  1-3; 1-6

- Hazardi tipa WAW ne postoje u ovoj sekvenci.
- U sekvenci postoji samo jedan ***control***-hazard, a on je izmedju instrukcija:

  5-6

Zadatak 3

$$a = b + c$$

$$d = e - f$$

Rešenje

Originalni kod sa zaustavljanjem:

```
        LW    Rb,b
        LW    Rc,c
Stall → ADD   Ra,Rb,Rc
        SW    Ra,a
        LW    Re,e
        LW    Rf,f
Stall → SUB   Rd,Re,Rf
        SW    Rd,d
```

Preuredjenni kod bez zaustavljanja

```
LW    Rb,b
LW    Rc,c
LW    Re,e
ADD   Ra,Rb,Rc
LW    Rf,f
SW    Ra,a
SUB   Rd,Re,Rf
SW    Rd,d
```

Zadatak 4

Polazni program:

```
Loop:LD  F0, 0(R1)
     ADD   F4, F0, F2
     SD 0(R1), F4
     SUBI  R1, R1, #8
     BNEZ  R1, Loop
```

Zadatak 4 – Rešenje prva varijanta

Polazni program:

```
Loop:   LD   F0, 0(R1)
        ADD  F4, F0, F2
        SD   0(R1), F4
        SUBI    R1, R1, #8
        BNEZ    R1, Loop
```

```
        LD   F0, 0(R1)
Loop:   ADD  F4, F0, F2
        SD   0(R1), F4
        SUBI    R1, R1, #8
        BNEZ    R1, Loop
        LD   F0, 0(R1)
```

U **prozor zakašnjenog grananja** postavljamo instrukciju sa ciljne adrese grananja, tj. instrukciju na labeli Loop.

Zadatak 4 – Rešenje II varijanta

Polazni program:

```
Loop:   LD   F0, 0(R1)
        ADD  F4, F0, F2
        SD   0(R1), F4
        SUBI    R1, R1, #8
        BNEZ    R1, Loop
```

```
Loop:   LD   F0, 0(R1)
        ADD  F4, F0, F2
        SUBI    R1, R1, #8
        BNEZ    R1, Loop
        SD   8(R1), F4
```

Pošto u originalnom program SUBI dekrementira sadržaj R1 za 8, a u preuređenom program se SD postavlja nakon SUBI, potrebno je modifikovati adresu na koju SD pamti.

Zadatak 5
Zakašnjeni load i preuređenje instrukcija da bi se izbegli zastoji

- EX faza za instrukcije ADD i SUBD traje 2 ciklusa (instrukcije rade sa podacima u pokretnom zarezu). Pretpostaviti da se koristi premošćavanje.
- Da bi se nakon ovih instrukcija izbegli zastoji neophodno je program preurediti tako da iza ADD, tj. SUBD ne slede odmah instrukcije koje koriste njihove rezultate (u ovom primeru to su SD instrukcije).

Polazni program:

```
LD   F0, 0(R1)
ADD F4, F0, F2
SD   F4, 0(R1)
LD   F6, -8(R1)
SUBD F8, F6, F2
SD   F8, -8(R1)
```

Zadatak 5 - Rešenje

Polazni program:

```
LD   F0, 0(R1)
ADD F4, F0, F2
SD   F4, 0(R1)
LD   F6, -8(R1)
SUBD    F8, F6, F2
SD   F8, -8(R1)
```

Preuređeni program:

```
LD   F0, 0(R1)
LD   F6, -8(R1)
ADD F4, F0, F2
SUBD F8, F6, F2
SD   F4, 0(R1)
SD   F8, -8(R1)
```