|  |  |
| --- | --- |
| Description: title-left | **ELEKTROTEHNIČKI FAKULTET**  BEOGRAD |

**Projektovanje 32-bitnog procesora opšte namene**

student: Jelena Živković

0251/2012

student: Stefan Tomić

0340/2012

profesor: dr Veljko Milutinović

asistent: Živojin Šuštran

|  |  |
| --- | --- |
| broj osvojenih poena: |  |

Beograd

Septembar 2016.

**Projektovanje 32-bitnog procesora opšte namene**

Jelena Živković, StefanTomić

*email:*[*zj120251d@student.etf.rs*](mailto:zj120251d@student.etf.rs)*,* [*ts120340d@student.etf.rs*](mailto:ts120340d@student.etf.rs)

**1. DEFINISANJE PROJEKTA**

* 1. **Uvod**

Projektuje se 32-bitni procesor opšte namene. Procesor je povezan sa 2 keš memorije, jednom za podatke i jednom za instrukcije. Interfejs procesora prema okolini sadrži linije za komunikaciju sa keš memorijama, RESET signal I signal kloka. Linije za komunikaciju sa keš memorijom podataka su 1) 32adresne linije, 2) 32 linije za čitanje podataka, 3) 32 linije za upis podataka I 4) kontrolne linije. Adresibilna jedinica je reč veličine 4 bajta.

Procesor sadrži jednu ALU jedinicu i jednu branch jedinicu. Jezgro je RISC arhitektura.

Procesor ima i protočnu obradu u 5 stepena.

Hazardi se razrešavaju hardverski, a postoji predikcija skoka i prosleđivanje. Prediktor skoka je realizovan po principu keša sa dvobitnom šemom.

**1.2 Ciljevi projekta**

Cilj projekta je projektovanje 32-bitnog procesora korišćenjem VHDL jezika za opis hardvera. Krajnji cilj je da se opis procesora može sintetizovati i da se prikaže simulacija svih funkcionalnosti.

**2. OPIS DIZAJNA**

**2.1 Zabeleške uz dizajn**

Uvedene pretpostavke zasnovane su na klasičnoj arhitekturi pipeline procesora rađenoj na predmetu Arhitektura i organizacija računara. Prediktor za skokove realizovan je asocijativno. Početna vrednost predikcije za sve skokove TAKEN. U ID fazi se ustanovljava da li dolazi do hazarda i onda se prosleđuju odgovarajući signali.

**2.2 Faze dizajna**

**IF faza**

U IF fazi se dohvata instrukcija iz instrukcione memorije na osnovu PC-a. Ovde se takođe i određuje vrednost registra PC koja će biti upotrebljena kako bi se dohvatila instrukcija. Vrednost PC-a se određuje na osnovu tipa instrukcije. Ukoliko je u pitanju instrukcija skoka vrednost će biti ona na koju treba da skočimo, a ukoliko nije vrednost će biti jednaka trenutnoj vrednosti uvećanoj za 1. Kako bi se odredilo koja je instrukcija u pitanju IF fazu je ugrađen prediktor skoka pomoću kojeg se dohvata adresa destinacije skoka ukoliko je pretpotstavka da je skok TAKEN. Ukoliko nam je predikcija loša, to će se utvrditi u nekoj od narednih faza.

**ID faza**

U ID fazi dekodujemo instrukciju koja smo dohvatili u prethodnoj fazi. Ovde se takodje pristupa registarskom fajlu oi određuje koji su adresirani registri i neposredne vrednosti zapisane u instrukciji (ako instukcija koristi iste). Paralelno sa čitanjem iz registarskog fajla , se nižih 16 razreda registra IR, koji sadrže 16 bitova neposredne veličine, se u proširuju na 32bitnu veličinu.

Na osnovu poređenja registara koji se koriste u instrukcijama iz sledećih faza pipeline-a ustanovljavamo da li je došlo do hazarda. U ovoj fazi se razrešavaju odgovarajući hazardi prosleđivanjem.

**EX faza**

U EX fazi realizovana je ALU jedinica koja izvršava, u zavisnosti od instrukcije u registru IR, jednu od aritmetičko-logičkih instrukcija . Takođe, u njoj se određuje da li su uslovi skokova ispunjeni

**MEM faza**

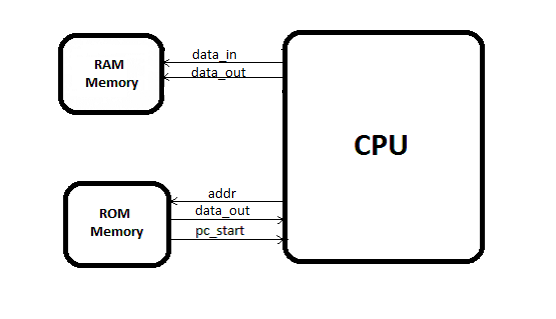
U MEM fazi se izvršavaju dve vrste instrukcija. Prva vrsta su instrukcije koje rade sa memorijom – LOAD i STORE, a druga instrukcije koje rade sa stekom – PUSH i POP.

U ovoj fazi se ,takođe, postavlja I signal za pražnjenje piprline-a (flush), ukoliko je prediktovana adresa skoka bila neodgovarajuća.

**WB faza**

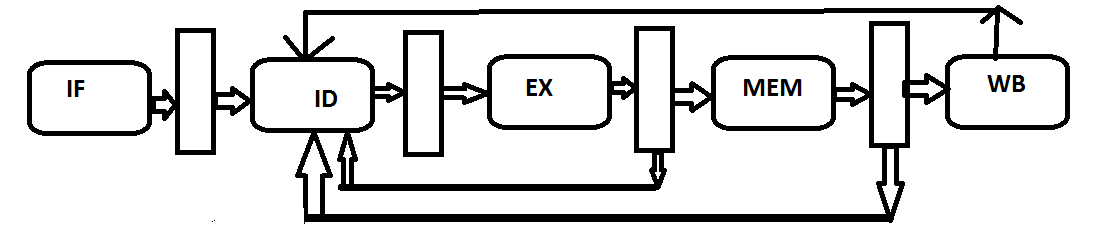
U ovoj fazi se vrši upis u registarski fajl, ukoliko je instrukcija koja se nalazi u ovoj fazi neka od instrukcija koja upisuje u registar.U registarski fajl se upisuje vrednost očitana iz memorije.

## 2.2.1 Interfejs između procesora i memorije



**2.2.2 Faze izvršavanja instrukcije**

Realizacija međufazne komunikacije prikazana je na skici ispod. U okviru faza IF, ID, EX i MEM nalaze se registri IF/ID, ID/EX, EX/MEM i MEM/WB, respektivno. Ovi registri služe za prenos podataka između faza pipeline-a.



**3. IMPLEMENTACIJA**

**3.1 Hazardi**

U ovakvoj realiziciji procesora. sa razdvojenom memorijom za podatke i memorijom za instrukcije nemoguće je da dođe do pojave strukturalnih hazarda. Dakle, javljaju se samo hazardi podataka ili upravljački hazardi koji se razrešavaju prosleđivanjem ili zaustavljanjem pipeline-a.

**3.2 Oporavak od hazarda**

Hazarde razrešavamo u ID fazi pomoću f-je decod\_instr u kojoj se dekoduje instrukcija, a zatim na osnovu toga da li je hazard u destinacionom registru upisuje odgovarajuća prosleđena vrednost u taj registar.

**3.3 Primeri hazarda**

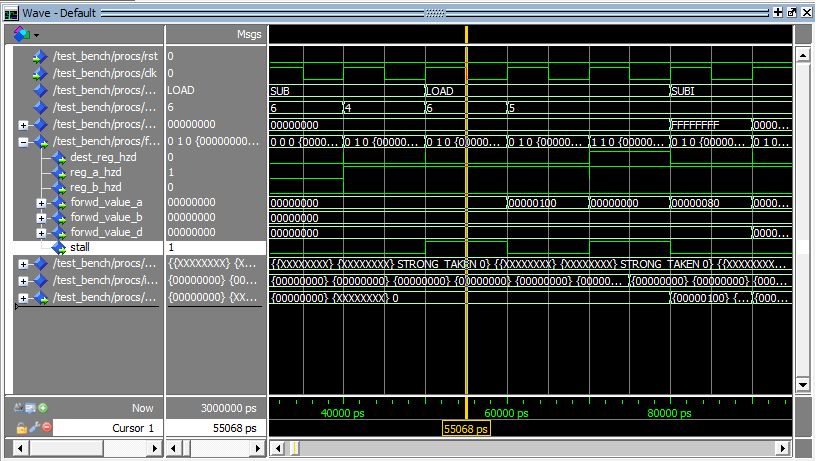
Na primeru iz javnog testa biće pokazano kako ovaj procesor razrešava hazarde.

00001000 00100100110001100011000000000000 SUB R6, R6, R6

00001001 00100100100001000010000000000000 SUB R4, R4, R4

00001002 00000000110001100000000000000000 LOAD R6,R6

00001003 00000000101001100000000000000000 LOAD R5,R6

****

**3.4 Prediktor skoka**

Uveden je prediktor koji je realizovan kao FIFO keš. Uzeto je da je početna vrednost predikcije skoka TAKEN. U EX fazi dolazi do provere uslova skoka i poređenja sa prediktovanom vrednošću na osnovu čega se odlučuje da li će se pipeline flush-ovati ili ne. Signal isFull se generiše I flushing se radi u MEM fazi. Ukoliko je potrebno, u prediktoru se ažurira ulaz ili pravi novi, tako da prilikom sledećeg čitanja instrukcije sa te adrese se čita iz keša i prosleđuje IF fazi.

**4. TESTIRANJE I VERIFIKACIJA**

Simulacija rada procesora rađena je korišćenjem alata ModelSim. U te svrhe korišćeni je testbench koji testira sve instrukcije i koji testira sve hazarde. Takođe, za proveru korektnosti rada procesora korišćen je i javni test dostupan na sajtu.

Svaki od testova sastoji se iz tri fajla, jedan koji sadrži instrukcije (učitava se u instrukcijsku keš memoriju – fajl instr\_cache), drugi koji sadrži početni sadržaj memorije za podatke (preslikava se u keš memoriju za podatke – data\_cache) i fajl u kome je dato očekivano stanje memorije nakon pravilnog izvršavanja instrukcija datih u prvom fajlu.

**4.1. Javni test**

Za potrebe testiranja korektnosti rada procesora korišćen je iavni test dat u fajlu javni\_test\_inst\_in.txt, a dostupan je na sajtu predmeta.

NAPOMENA: Za naredne testove naveden je samo asemblerski kod, ne i sadržaji memorija instrukcija I podataka.

**4.2 Test 1**

Sledeći test proverava kako procesor razrešava hazarde.

XOR R6 R6 R6

XOR R7 R7 R7

XOR R8 R8 R8

XOR R9 R9 R9

XOR R10 R10 R10

XOR R11 R11 R11

MOVI R8 100

MOVI R10 1

MOVI R7 8

BEQ R6 R7 6

LOAD R9 R8 0

AND R9 R9 R10

ADD R11 R11 R9

ADDI R8 R8 1

SUBI R7 R7 1

JMP R6 109

STORE R6 R11 100

XOR R2 R2 R2

XOR R3 R3 R3

HALT

**4.3 Test 2**

U sledećem testu se vrši sumiranje elemenata niza.

MOVI R5 256

MOVI R6 512

XOR R7 R7 R7

MOVI R8 10

XOR R9 R9 R9

PUSH R8

STORE R5 R8 0

LOAD R8 R6 0

ADD R6 R6 R7

ADDI R6 R6 1

LOAD R8 R5 0

SUBI R8 R8 1

STORE R5 R8 0

BNQ R9 R8 -7

POP R9

HALT

**4.4 Test 3**

U narednom testu se vrši oduzimanje dva niza smeštena u memoriji. Rezultat se smešta na adresu prvog niza.

A(i) = B(i) – A(i)

100

100 XOR R0 R0 R0

101 MOV R1 100h

102 MOV R2 200h

103 MOV R3 50h

104 LD R4 R3 0h

105 BEQ R4 R0 8h

106 LD R6 R1 0h

107 LD R7 R2 0h

108 SUB R8 R7 R6

109 STORE R1 R8 0h

10A SUBI R4 R4 1h

10B ADDI R1 R1 1h

10C ADDI R2 R2 1h

10D JMP R0 105h

10E ADDI R1 R1 1h

10F ADDI R1 R1 1h

110 HALT

**5. ZAKLJUČAK**

Razmatranje urađenog i ideje za dalji rad.

Da bi se razrešio problem pristupa memoriji pri čitanju instrukcija i pri čitanju podataka,odnosno da bi se što više ubrzao pristup i povećala efikasnost, kreirane su dve odvojene memorije. Kako bi se, pri instrukcijama skoka uštedeli taktovi, uveden je predictor u IF fazu. Kako bi se ubrzala protočna obrada hazardi su razrešavani hardverski i to ili prosleđivanjem ili zaustavljanjem protočne obrade. Zaključeno je da je hardverska paralelizacija na visokom nivou.

**6. LITERATURA**

[1] http://home.etf.rs/~vm/os/vlsi/index.html

[2] http://rti.etf.bg.ac.rs/rti/ri3aor/literatura/predavanja/

[3] <https://www.altera.com/downloads/download-center.html>

[4] Pipeline, Jovan Đorđević