Protočnost

Kontrolni hazardi (Control hazards)

kontrolni hazardi

- Mogu uzrokovati veći gubitak performansi nego hazardi po podacima.
- Nastupaju zbog instrukcija koje mogu promeniti sadržaj PC (branch, jump, call, return).
- Primer branch instrukcije: novi sadržaj PC poznat tek u MEM fazi, posle izračunavanja adrese i testiranja uslova.
- Neophodno zaustaviti protočni sistem dok se ne dozna novi sadržaj PC.

branch	IF	ID	EX	MEM	WB
i+1		IF	--	--	IF

zaustavljanje protočnog sistema nije moguće odmah nakon pribavljanja branch jer nije završeno dekodiranje. Pribavljena instr. se briše (IF/ID registar) Nakon Mem faze vrši se novo pribavljanje

Kako redukovati gubitke?

- Gubljenje 3 clk ciklusa za svaku naredbu grananja znatno degradira performanse sistema (1/6 svih nardbi u programu su branch).
- Da bi se redukovali gubitci može se učiniti sledeće:
  - ustanoviti što ranije da li dolazi do grananja
  - za slučaj grananja, što ranije odrediti novi sadržaj PC
- Obe stvari potrebno je odrediti što ranije u protočnom sistemu.
- Ako naredba grananja vrši samo testiranje na nulu (BEQZ ili BNEZ) tada se testiranje može obaviti u ID fazi kada se pristupa RF.
- Novi sadržaj PC je takodje moguće odrediti na kraju ID faze.
- Zahteva se dodatni sabirač, jer je ALU zauzet u EX fazi (ove dve faze se preklapaju)

Polazna staza podataka

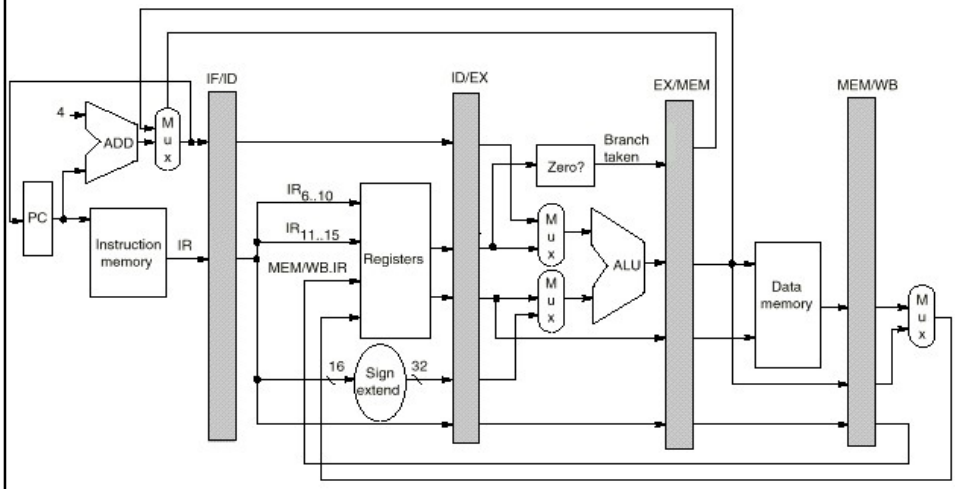
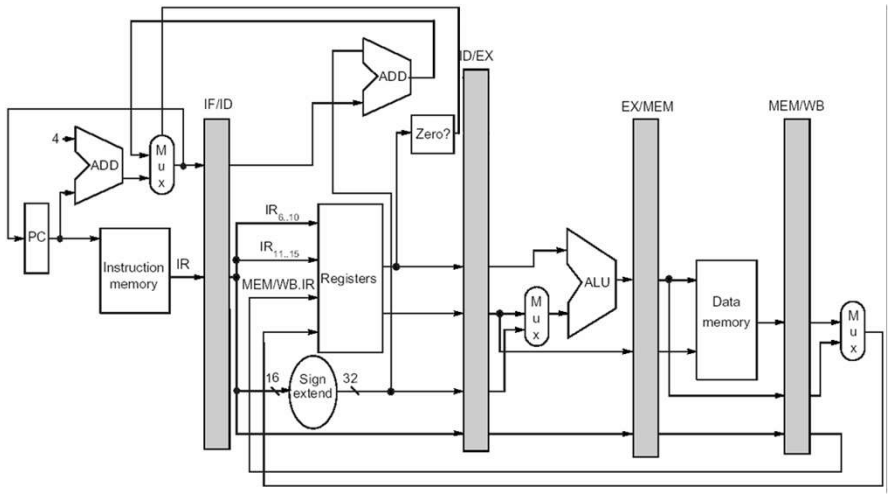


FIGURE 3.4 The datapath is pipelined by adding a set of registers, one between each pair of pipe stages.

Modifikovana staza podataka



- Redukovanje kašnjenja zbog branc hazarda pomeranjem testiranja na 0 i izračunavanjem adrese u ID fazi.

Modifikacija IF faze

- Instruction Fetch
  - IF/ID.IR ← mem[PC]
  - IF/ID.NPC, PC ← if (Regs[IF/ID. $IR_{6...10}$] op 0) {IF/ID.NPC +(IF/ID.IR_{16})^{16}##IF/ID.$IR_{16...31}$}else{PC+4}
- Dejstvo:
  - Pribavljanje instrukcije iz memorije.
  - Inkrementiranje PC za 4 da bi se pristupilo sledećoj instrukciji ili postavljanje nove vrednosti PC na osnovu adrese grananja odredjene u ID fazi prethodne instrukcije (branch)

Modifikacija ID faze

- Instruction Decode Cycle/Register Fetch
  - ID/EX.A ← Regs[$IR_{6...10}$]; ID/EX.B ← Regs[$IR_{11...15}$]
  - ID/EX.IR ← IF/EX.IR
  - ID/EX.Imm← (IF/ID.IR_{16})^{16}##IF/ID.$IR_{16...31}$
  - Izračunavanje uslova: Cond ← Regs[IF/ID.$IR_{6..10}$] op 0
  - Izaračunavanje ciljne adrese grananja: IF/ID.NPC + (IF/ID.IR_{16})^{16}##IF/ID.$IR_{16...31}$
  - Dejstvo
  - Dekodiranje instrukcije i pristupanje RF; Sadržaji registara opšte namene se pamte u privremene registre (A i B su deo protočnog regista ID/EX stepena)
  - Izdvajanje nepsrednog operanda i Smeštanje u Imm (deo ID/EX protočnog registra),
  - Izračunavanje uslova cond i ciljne adrese u slučaju da dodje do grananja da bi se postavila nova vrednost PC
- Faze EX, MEM i WB za branch su prazne

Šta je postignuto modidfikacijom?

- Smanjen je gubitak sa 3 na 1 clk ciklus.

branch	**IF**	**ID**	**EX**	**Mem**	**WB**
i+1		**IF**	**IF**	**ID**	**EX**

- Kompajlerskim tehnikama moguće je dalje redukovati kašnjenje:
  - Jedno rešenje je predvideti da se grananje neće obaviti i u tom slučaju se stanje mašine neće promeniti sve dok ne bude poznat ishod grananja.
    - Izvršenje se nastavlja sa sledećom instrukcijom kao da nije u pitanju naredba grananja.
    - Protočni sistem se zaustavlja ako grananje treba da se obavi. Prethodno pribavljena instrukcija se briše.

Ponašanje sistema sa predvidjanjem da se grananje neće obaviti

Untaken branch instruction	IF	ID	EX	MEM	WB				
Instruction $i + 1$		IF	ID	EX	MEM	WB			
Instruction $i + 2$			IF	ID	EX	MEM	WB		
Instruction $i + 3$				IF	ID	EX	MEM	WB	
Instruction $i + 4$					IF	ID	EX	MEM	WB

Taken branch instruction	IF	ID	EX	MEM	WB				
Instruction $i + 1$		IF	**idle**	**idle**	**idle**	**idle**			
Branch target			IF	ID	EX	MEM	WB		
Branch target + 1				IF	ID	EX	MEM	WB	
Branch target + 2					IF	ID	EX	MEM	WB

Druga mogućnost

- Predvideti da će se grananje obaviti i početi sa pribavljanjem instrukcije sa ciljne adrese grananja
  - Ovaj prilaz u našem primeru nema prednosti jer su uslov grananja i adresa grananja (ako do grananja doje) poznati u isto vreme
- Ovaj prilaz ima smisla u sistemima gde su uslovi koji se testiraju u instrukciji grananja složeniji, pa uslov može biti kasnije izračunat od adrese.

Na sonovu čega se obavlja predvidjanje?

- Postoje dva osnovna metoda da se statički predvidi grananje u fazi kompilacije:
  - Posmatranjem ponašanja programa i korišćenjem informacija dobijenih na osnovu prethodnih izvršenja programa.
    - Na primer, ponašanje programa može biti takvo da se većina grananja obavlja. U takvim slučajevima najjednostavnije je predvideti da će se sva grananja obaviti.
  - Predvideti grananje na osnovu smera grananja:
    - predvideti da se sva grananja u nazad (programske petlje) obavljaju
    - predvideti da se sva grananja u napred ne obavljaju.

Treća mogućnost – zakašnjeno grananje (delayed branch)

- Instrukcija koja sledi iza naredbe grananja se izvršava bez obzira da li će se grananje obaviti ili ne.
- Naziv potiče od činjenice da se efekat naredbe grananja odlaže.

conditional branch instruction
sequential successor$_1$
sequential successor$_2$
.....
sequential successor$_n$
branch target if taken

slot zakašnjenog grananja

- Za instrukcije koje slede iza naredbe grananja se kaže da se nalaze u slotu (prozoru) zakašnjenog grananja. Ove instrukcije se izvršavaju bez obzira da li dolazi do grananja ili ne.
- U praksi je veličina prozora najčešće 1.
- Zadatak kompajlera je da u ovaj prozor postavi važeće i korisne instrukcije

Efekat tehnike zakašnjenog grananja

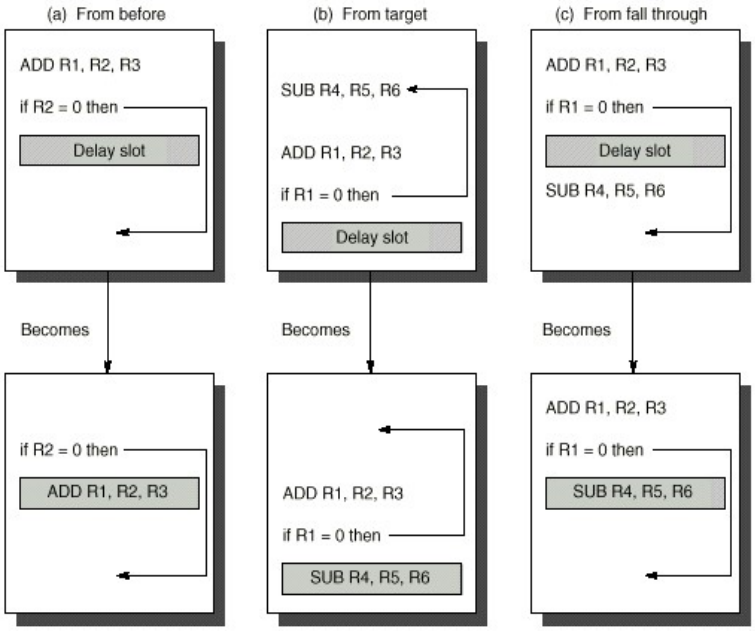
Untaken branch instruction	IF	ID	EX	MEM	WB			
Branch delay instruction (i + 1)		IF	ID	EX	MEM	WB		
Instruction i + 2			IF	ID	EX	MEM	WB	
Instruction i + 3				IF	ID	EX	MEM	WB
Instruction i + 4					IF	ID	EX	MEM	WB

Taken branch instruction	IF	ID	EX	MEM	WB			
Branch delay instruction (i + 1)		IF	ID	EX	MEM	WB		
Branch target			IF	ID	EX	MEM	WB	
Branch target + 1				IF	ID	EX	MEM	WB
Branch target + 2					IF	ID	EX	MEM	WB

Izbor instrukcije koja se postavlja u delay slot

* Postoje tri mogućnosti:
  - A Nezavisna instrukcija koja se u programu nalazi pre naredbe grananja:
    - Uvek dovodi do poboljšanja performansi. Instrukcija grananja ne sme da zavisi od instrukcije koja se postavlja u slot.
  - B Instrukcija sa ciljne adrese grananja:
    - Poboljšava performanse ako se grananje obavi. Može zahtevati dupliranje instrukcije koja se postavlja u slot. Izvršenje instrukcije ne sme uticati na korektnost programa ako se grananje ne obavi.
  - C Jedana od instrukcija koje se nalaze iza naredbe grananja:
    - Poboljšava performanse ako se grananje ne obavi. Izvršenje instrukcije ne sme uticati na korektnost programa ako se grananje obavi.

Primeri



Performanse sistema u prisustvu hazarda

- Ubrzanje protočnog sistema u prisustvu kontrolnih hazarda (usvajajući idealni CPI od 1)

$$\text{ubrzanje protočnog sistema} = \frac{\text{Dubina sistema}}{1 + \text{Broj izgubljenih ciklusa}}$$

Broj izgubljenih ciklusa zbog grananja = Učestalost grananja X kašnjenje zbog grananja

$$\text{ubrzanje protočnog sistema} = \frac{\text{Dubina sistema}}{1 + \text{Učestalost grananja X kašnjenje zbog grananja}}$$

Primer

- Usvojimo sledeći miks instrukcija :

Tip	učestalost pojavljivanja	
Arith/Logic	40%	
Load	30%	od čega u 25% slučajeva odmah sledi instrukcija koja koristi vrednost load
Store	10%	
branch	20%	od čega se 45% obavi

- Koliko iznosi CPI (srednji broj taktova po instrukciji) ako se koristi premošćavanje, odluka o grananju se donosi u ID i koristi se predvidjanje da se grananje neće obaviti ?

CPI = Ideal CPI + Kašnjenje usled hazarda po instrukciji

= 1 + kašnjenja od load + kašnjenja od branch

= 1 + 0.3 x 0.25 x 1 + 0.2 x 0.45 x 1

= 1 + 0.075 + 0.09

= 1.165

Dinamička predikcija grananja

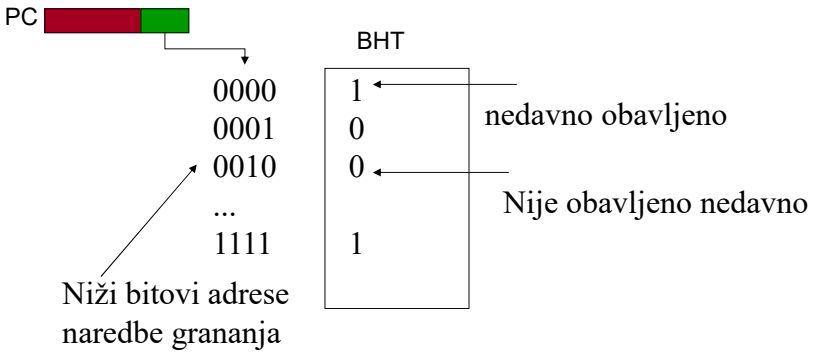
- Učestalost naredbi grananja u programima zahteva da se smanje potencijalni zastoji uzrokovani kontrolnim hazardima.
- Statičke šeme kojima se smanjuje uticaj kontrolnih hazarda obuhvataju
  - zakašnjeno grananje
  - odmotavanje petlje
- Dinamička predikcija grananja se razlikuje od statičke jer koristi ponašanje grananja u toku izvršenja programa da predvidi ishod grananja.
- Cilj svih tehnika za predikciju grananja je da omoguće procesoru da razreši ishod grananja i tako spreči da kontroli hazardi izazovu zastoje
- Efikasnost tehnika za predvidjanje ishoda grananja ne zvisi samo od pouzdanosti predvidjanja, već i od "cene" grananja kada je predvidjanje tačno i kada nije.

Dinamička predikcija grananja

- Neke od predloženih tehnika za dinamičku predikciju grananja su
  - **Jedno-nivovske ili bimodalne: koriste Branch History Table (BHT) ili Branch Prediction buffer (BPB), tabelu koja sadrži jedno ili dvo-bitne prediktore koji se indeksiraju nižim bitovima naredbe grananja.**
  - **Dvo-nivovski ili korelacioni prediktori**
  - **Baferi ciljne adrese grananja (Branch Target Buffer – BTB): sadrže adrese naredi grananja koje su se ostvarile zajedno sa adresom ciljne naredbe grananja**
  - **Hibridni prediktori: koriste kombinaciju dve ili više šema (obično dve) za predikciju grananja.**

Baferi za predikciju grananja (Branch prediction buffer –BPB ili Branch History Table –BHT)

- BPB je mala memorija indeksirana nižim bitovima adrese naredbe grananja
- Najjednostavnija varijanta koristi samo **jedan** bit za predikciju koji ukazuje da li se grananje nedavno obavilo (**1**) ili ne (**0**)



BHT (ili BPB)

- Najjednostavnija šema
- Od koristi je u sistemima kod kojih je adresa grananja poznata ranije od uslova grananja
- BHTse pristupa u ID fazi (kada se obavi dekodiranje)
  - S obzirom da se za indeksiranje BHT koriste niži bitovi (a ne cela) adrese naredbe grananja, predvidjanje koje se koristi može biti i od neke druge naredbe grananja koja je imala iste niže bitove adrese.
- Pibavljanje nove instrukcije počinje od predvidjenog pravca
- Efikasnost tehnike zavisi od toga koliko često je reč o branch instrukciji koja je od interesa, i koliko je predvidjanje tačno kada se ostvari uparivanje

BHT – problemi

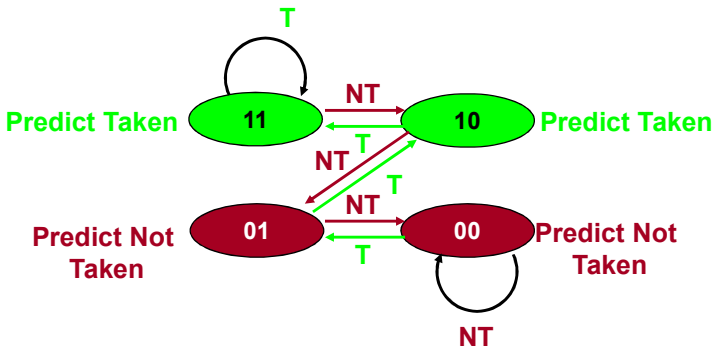
- Čak i ako se grananje skoro uvek obavlja (petlje), ova šema će kod ugnježđenih petlji imati dva pogrešna predvidjanja (umesto jednom)

```
Loop1
        Loop2

    end Loop2

end Loop1
```

• Predvidjanje će biti pogrešno kod prve i poslednje iteracije Loop2:
  •pogrešno predvidjanje kod poslednje iteracije je neizbežno (jer će bit biti postavljen na 1 – prethodno grananje se obavilo
  •Pogrešno predvidjanje kod prve iteracije nastaje jer je bit bio postavljen na 0 prethodnim izvršenjem poslednje iteracije petlje, jer se grananje nije obavilo u toj iteraciji

BHT – 2-bitna šema

- Da bi se poboljšala pouzdanost predvidjanja umesto 1-bitne koristi se 2-bitna šema.
- Kod 2-bitne šeme predvidjanje mora biti dva puta pogrešno da bi se promenila predikcija



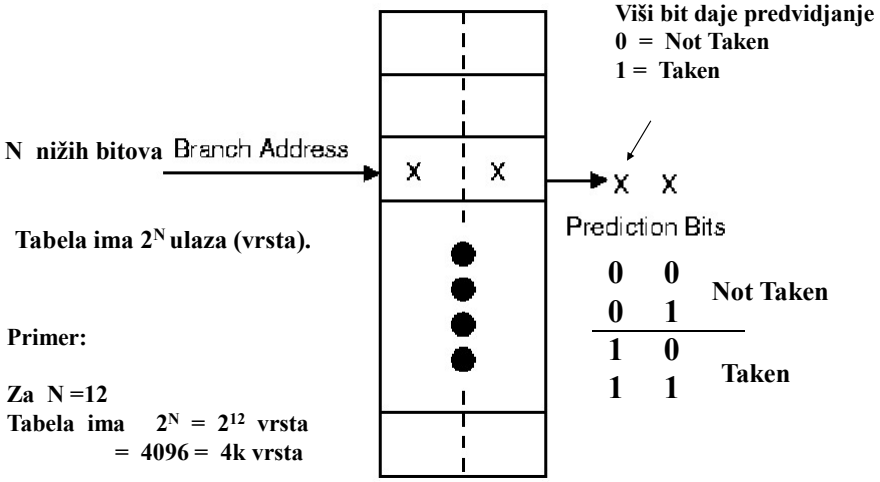


Tabela ima 2^N ulaza (vrsta).

Primer:

Za N =12
Tabela ima $2^N = 2^{12}$ vrsta
= 4096 = 4k vrsta

Broj potrebnih bitova = 2 x 4k = 8k bitova

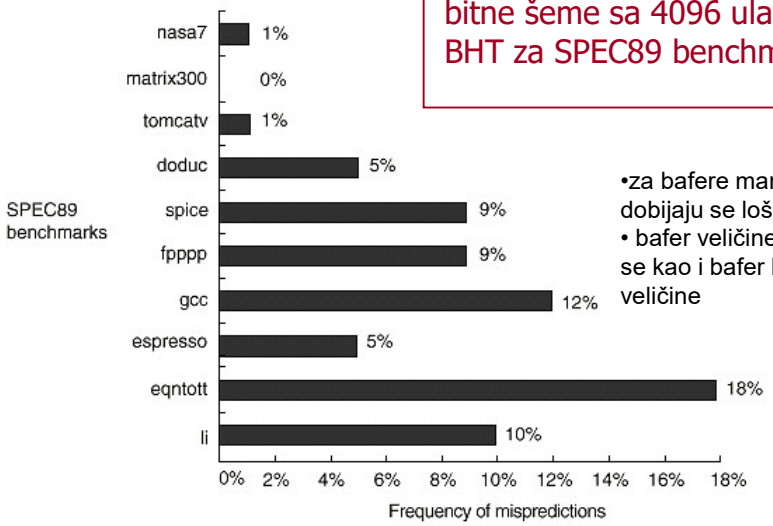
n-bitni prediktori

- 2-bitni prediktor je specijalni slučaj n-bitne šeme koja koristi n-btni zasićeni brojač za predikciju grananja.
- Vrednosti koje brojač može uzeti su u opsegu 0-2^n-1.
- Ako je vrednost brojača $\geq 2^{n-1}$ predvidjanje je da će se grananjae obaviti (taken), u suprotnom da se ne obavlja (not taken).
- Brojač se inkrementira svaki put kada se grananje obavi, a dekrementira kada se grananje ne obavi
  - Kada je vrednost brojača 2^n-1, nema inkrementiranja
  - Kada je vrednost brojača 0, nema dekrementiranja
- Proučavanja n-bitnih prediktora su pokazala da se 2-bitni prediktori ponašaju skoro isto dobro kao n-bitni (n>2) prediktori.
  - zbog toga većina sistema koristi 2-bitne prediktore

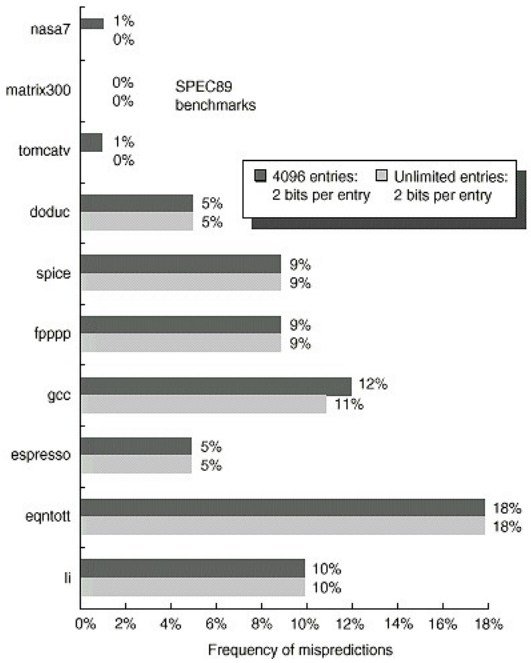
BHT (nast.)

- Predikcioni baferi se mogu implementirati kao mali, specijalni, keš kome se pristupa u toku ID faze.
- Ako se dekodira branch instrukcija i ako je predvidjanje da će se grananje obaviti, pribavljanje nove instrukcije počinje čim je poznat sadržaj PC
- Ako je predvidjanje da se grananje neće obaviti, nastavlja se sa sekvencijalni pribavljanjem i izvršenjem
- Ako je predvidjanje bilo pogrešno, predikcioni bitovi se menjaju, pribavljena instrukcija se poništava i pribavlja korektna instrukcija

Pouzdanost predvidjanja 2-bitne šeme sa 4096 ulaza u BHT za SPEC89 benchmark



- za bafere manje veličine dobijaju se lošiji rezultati
- bafer veličine 4K ponaša se kao i bafer beskonačne veličine

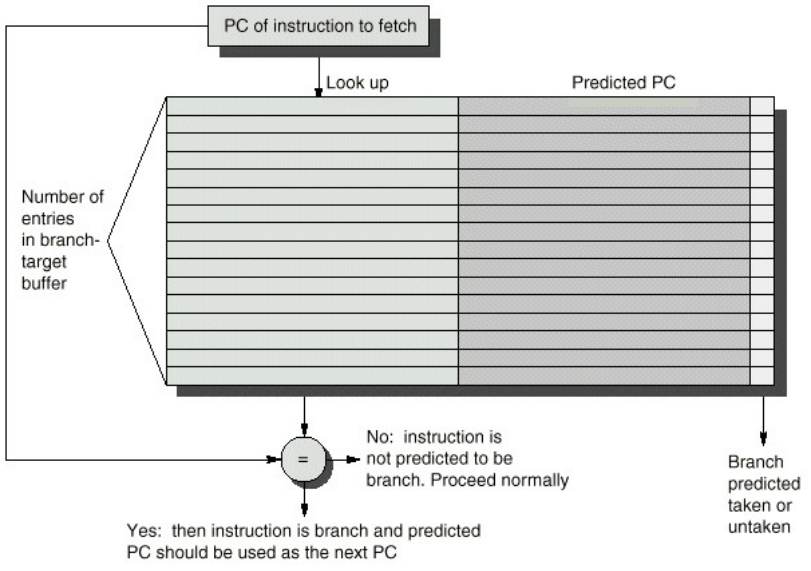


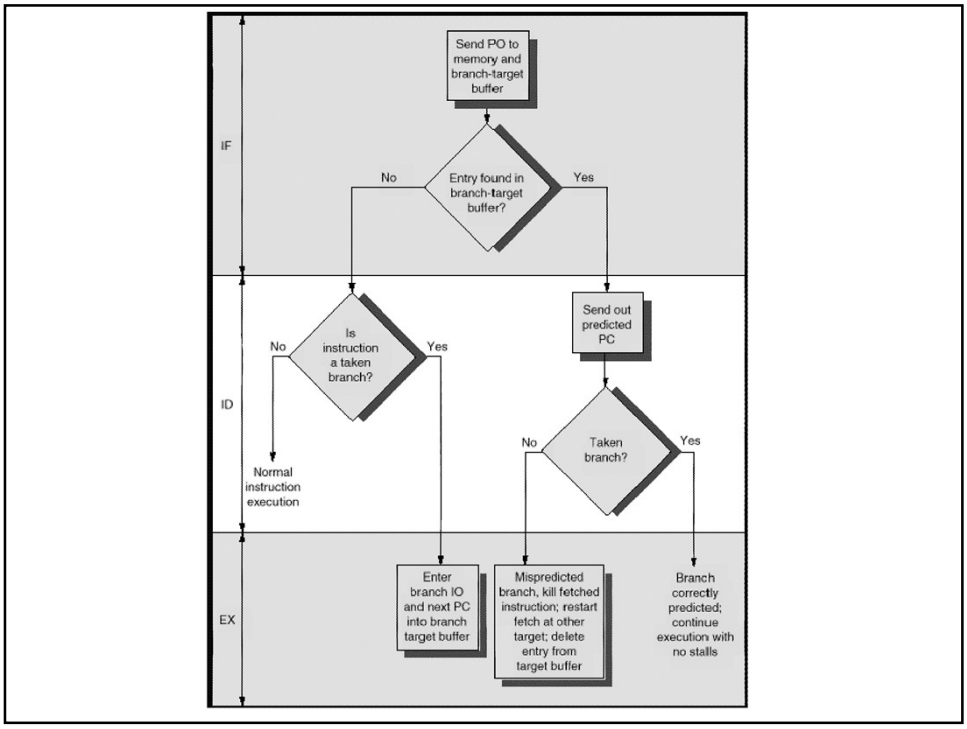
Pouzdanost predvidjanja BHTsa 4096 ulaza i sa neograničenim brojem ulaza sa 2-bitnom šemom za SPEC89

Baferi ciljne adrese grananja (Branch Target Buffers – BTB)

- Da bi se redukovali zastoji zbog naredbi grananja kod analiziranog 5-to stepenog RISC procesora potrebno je znati sa koje adrese treba pribaviti instrukciju na kraju IF faze.
- To znači da moramo znati da li je još nedekodirana instrukcija instrukcija grananja, i ako jeste, koji je sledeći sadržaj programskog brojača.
  - na taj način bi se zastoji uzrokovani naredbom grananja sveli na 0
- Bafer koji pamti predvidjene adrese za sledeću instrukciju nakon naredbe grananja zove se Branch Target Buffer – BTB ili branch target cache
- BTB se pristupa u toku IF faze, korišćenjem adrese pribavljene instrukcije (moguće branch) da bi se pristupilo baferu:
  - Ako postoji pogodak, onda se zna adresa sledeće instrukcije na kraju IF faze, što je 1 clk ciklus ranije nego kod BHT (BPB)

BTB





Gubitci(penali) kod BTB

Instruction in buffer	Prediction	Actual branch	Penalty cycles
Yes	Taken	Taken	0
Yes	Taken	Not taken	2
No		Taken	2

•PRIMER: Odrediti srednje gubitke u clk za nardbu grananja ako se koristi BTB usvajajući da je:

- pouzdanost predvidjanja 90%
- stopa pogotka u BTB 90%
- grananja se dešavaju u 60% slučajeva

•ODGOVOR:

gubitci = stopa_pogotka_u_BTB *procenat_pogrešnih_predikcija *2 + (1- stopa_pogotka_u_BTB) * procenat_obavljenih grananja *2

= 0.9*0.1*2 + 0.1*0.6*2 = 0.3 clk

Gubitci zbog naredbi grananja su svedeni na 0.3 clk