Credits: Asanovic @ MIT

Branch Prediction

Handout 16

November 16, 2004

Shoukat Ali

shoukat@umr.edu

UMR UNIVERSITY OF MISSOURI-ROLLA
The Name. The Degree. The Difference.

1

The Problem

- pipeline must be fed a new instruction every cycle for full performance
- what instruction should be fed in pipeline after a branch?
- control-transfer instructions require insertion of bubbles in the pipeline
- number of bubbles depends upon the number of cycles it takes
  - to determine the next instruction address, and
  - to fetch the next instruction

2

Number of Control Bubbles – Jump/Call

Target known after ID

	Clock cycle					
Instruction	1	2	3	4	5	6
Jump or call	IF	ID	EX	WB		
$i+1$		IF	IF	ID	EX	. . .
$i+2$			*stall*	IF	ID	. . .
$i+3$				*stall*	IF	. . .

Figure S.43 Effects of a jump or call Instruction on the pipeline.

PC+4 ("not-taken" path) fetched automatically.

After ID of jump/call, target is known so fetch is restarted. One cycle of "work" is wasted.

3

Number of Control Bubbles – Taken Branch

1. Target known after ID
2. Direction known after EX

	Clock cycle					
Instruction	1	2	3	4	5	6
Taken branch	IF	ID	EX	WB		
$i+1$		IF	*stall*	IF	ID	. . .
$i+2$			*stall*	*stall*	IF	. . .
$i+3$				*stall*	*stall*	. . .

Figure S.44 Effects of a taken conditional branch on the pipeline.

1

3

2

PC+4 ("not-taken" path) fetched automatically.

br pred removes this stall

After ID of branch, target is known BUT branch direction is not known. After branch dir is known at end of EX, fetch is restarted to get branch target. One cycle of "work" is wasted + 1 stall.

4

Number of Control Bubbles – Not Taken Branch

1. Target known after ID
2. Direction known after EX

	Clock cycle					
Instruction	1	2	3	4	5	6
Not-taken branch	IF	ID	EX	WB		
$i+1$		IF	*stall*	ID	EX	. . .
$i+2$			*stall*	IF	ID	. . .
$i+3$				*stall*	IF	. . .

Figure S.45 Effects of a not-taken conditional branch on the pipeline.

1

3

2

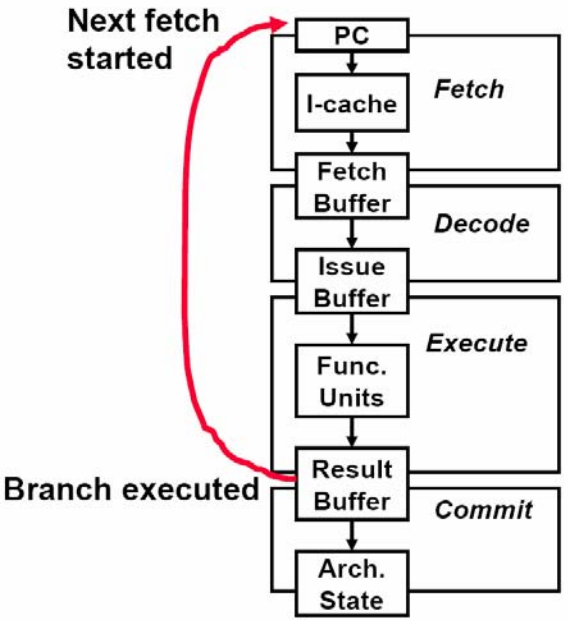
PC+4 ("not-taken" path) fetched automatically.

br pred removes this stall

After ID of branch, target is known BUT branch direction is not known. After branch dir is known at end of EX, fetch is NOT restarted because correct instr is already fetched. One stall.

5

Branch Penalty for Deeper Pipelines



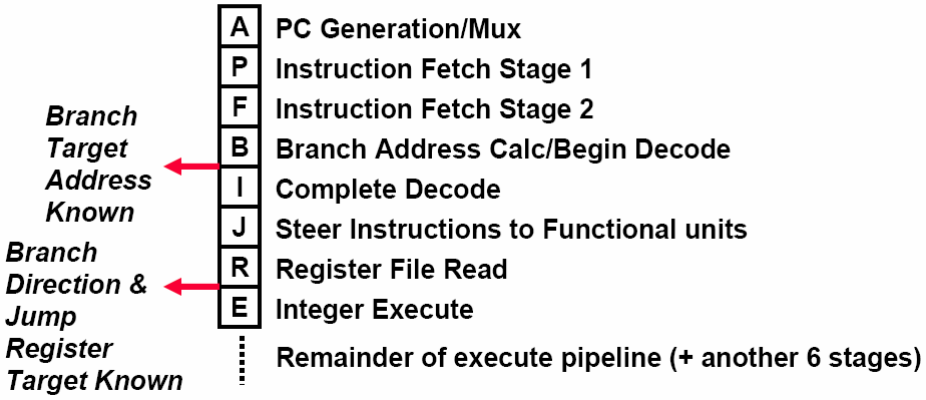
branch penalty = number of bubbles that must be inserted in the pipeline after a branch

also called branch delay slots

6

Branch Penalties in Modern Pipelines

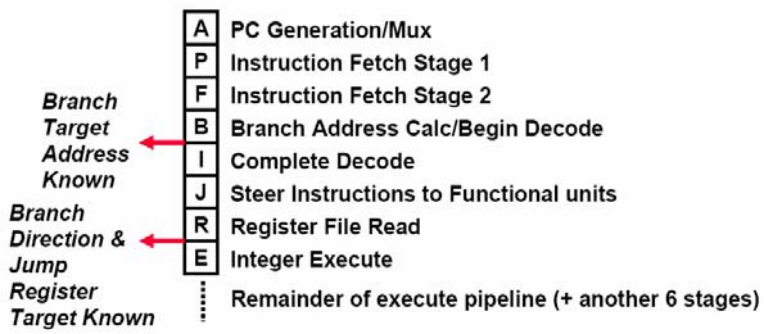
UltraSPARC-III instruction fetch pipeline stages
(in-order issue, 4-way superscalar, 750MHz, 2000)



NOTE A: branch target is the address of instruction on ***TAKEN PATH*** ;
NOTE B: for NOT TAKEN PATH, address is, of course, PC + 4

7

Branch Prediction



- if we predict the branch direction, next instruction can be fetched after stage B rather than after stage R
- modern branch predictors have high accuracy (>95%) and can reduce branch penalties significantly
- very helpful for deeply pipelined processors
  - is it helpful for our 5-stage pipeline with branch condition check done in ID? (see slide 11 of handout 6)

8

Hardware Support Required for Branch Prediction

1. prediction structures
   - branch prediction buffers (aka branch history tables), branch target buffers etc.

2. misprediction recovery mechanisms
   - in-order machines: squash instructions following a branch in pipeline
   - out-of-order machines: ROB

9

Types of Branch Prediction

- static
  - prediction does not depend on the recent history of the branch
  - no further discussion about static

- dynamic
  - prediction depends on the recent history of the branch

10

Dynamic Branch Prediction

- hardware predicts the branch to be taken or not taken
- prediction depends on the recent history of this branch
  - local predictor
- sometimes prediction depends on the recent history of past few branch instructions
  - global predictor or co-relating predictor
- Alpha 21264 uses both local and global predictor and then uses a tournament scheme to pick the better predictor
  - most sophisticated branch prediction scheme until 2001

11

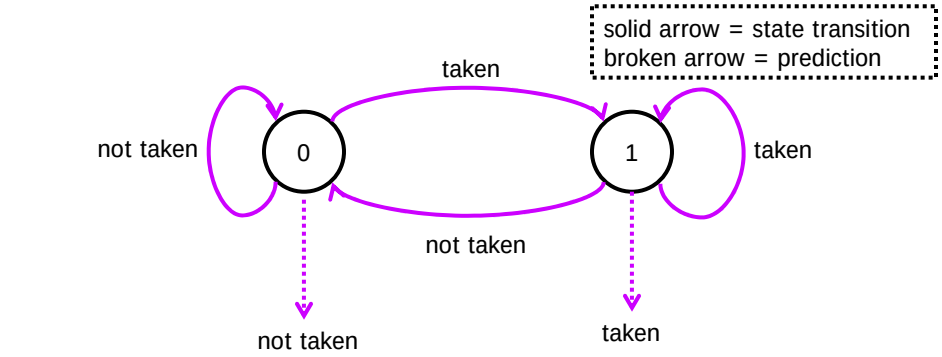
Need for Branch Prediction in Superscalar

- branches arrive N times faster in an N-way SS
- need for ambitious scheduling
  - in simple pipelines, a branch would cause 1 cycle stall, i.e. next instr cannot be issued without stalling for a cycle
  - in N-way SS pipeline, a branch could stall {(N-1) + N} next instructions
  - this motivates the need to develop excellent branch prediction schemes PLUS speculative execution methods (ROB)

12

1 Bit Saturating Counter

- 1 bit saturating counter – increments when branch is taken, decrements when not taken
  - does not reset; instead it saturates



if prediction bit = 0, counter predicts branch will be “not taken”

if prediction bit = 1, counter predicts branch will be “taken”

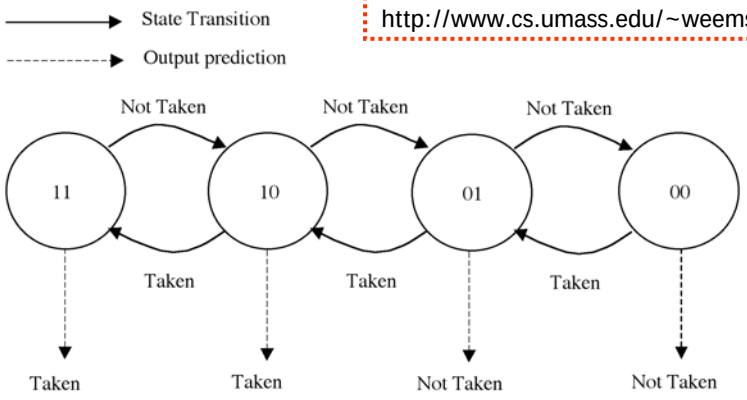
13

2 Bit Saturating Counter

- a two bit counter that can never go below 0 or above 3
- increment the counter when branches are taken
  - but saturate at 3
- decrement when branches are not taken
  - but saturate at 0
- to generate a prediction, examine the value of the counter
  - if it is 2 or 3, the branch is predicted to be taken
  - if it is 0 or 1, the branch is predicted to be not taken.

14

2 Bit Saturating Counter – Needs Convincing



if prediction bit (i.e., MSB) = 0, counter predicts branch will be "not taken"

if prediction bit (i.e., MSB) = 1, counter predicts branch will be "taken"

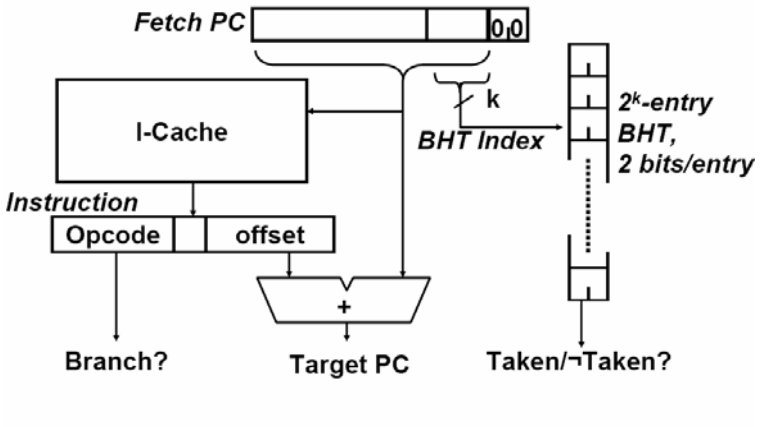
15

Branch Prediction Buffer (aka Branch History Table)

- prediction counters are stored in a table called branch prediction buffer (also known as branch history table, BHT)
- ideally there should be one counter for each branch, but that would mean a very large BHT
- typically a single counter is shared among many branches
  - +ve: much smaller BHT
  - -ve: counter trained by one branch might be used by another
- studies show that:
  - prediction accuracy of a 4096 entry BHT with a 2-bit counter/entry is very high
    - ~90-95% correct predictions (~5-10% mis-predictions)
  - a 4096 entry BHT is "as good as" an infinite entry BHT

16

Branch Prediction Buffer (aka Branch History Table)



4K-entry BHT, 2 bits/entry, ~80-90% correct predictions

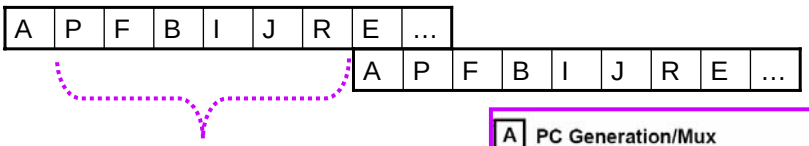
notes:
(1) Target PC is computed in parallel with decode (after fetch). This, BTW, happens even if we do not use branch prediction.

(2) BHT is checked in parallel with decode and branch target calculation

17

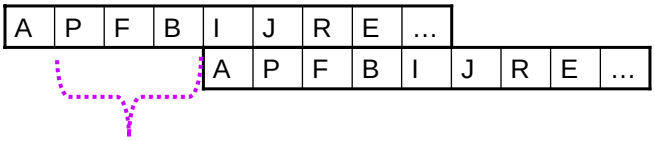
Advantage of BHT: UltraSPARC-III Fetch Pipeline

- reduces branch penalty for a taken branch from 6 to 3



6 CC penalty for correctly predicted branch

A	PC Generation/Mux
P	Instruction Fetch Stage 1
F	Instruction Fetch Stage 2
B	Branch Address Calc/Begin Decode
I	Complete Decode
J	Steer Instructions to Functional units
R	Register File Read
E	Integer Execute

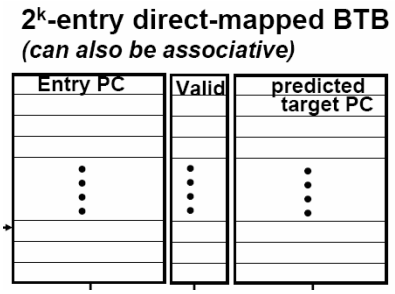


3 CC penalty for correctly predicted branch

18

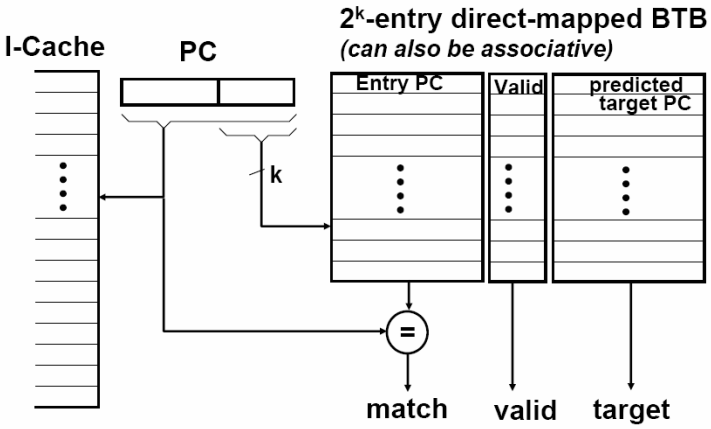
Complaint With BHT

- why can it not reduce branch penalty even lower?
- branch target buffer (BTB) does that
  - a cache of both the branch PC and target PC
  - only taken branches and jumps held in BTB
  - -ve: always assumes a branch is taken



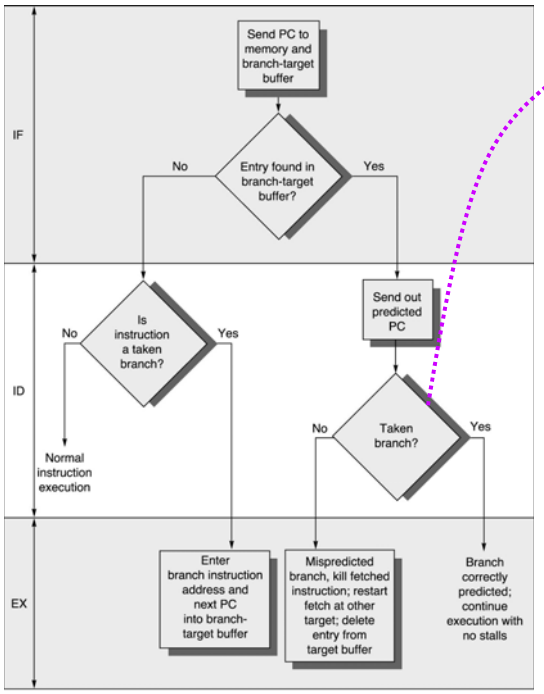
19

Branch Target Buffer



- *note: instr $ NOT involved in this operation (unlike BHT)*
- if match = 1, instruction is branch and predicted PC should be used as the next PC
  - else: not a branch or an untaken branch, next PC = PC + 4
- next PC determined ***in parallel*** with branch ***fetch***!!
  - in BHT, next PC determined ***after fetch***.

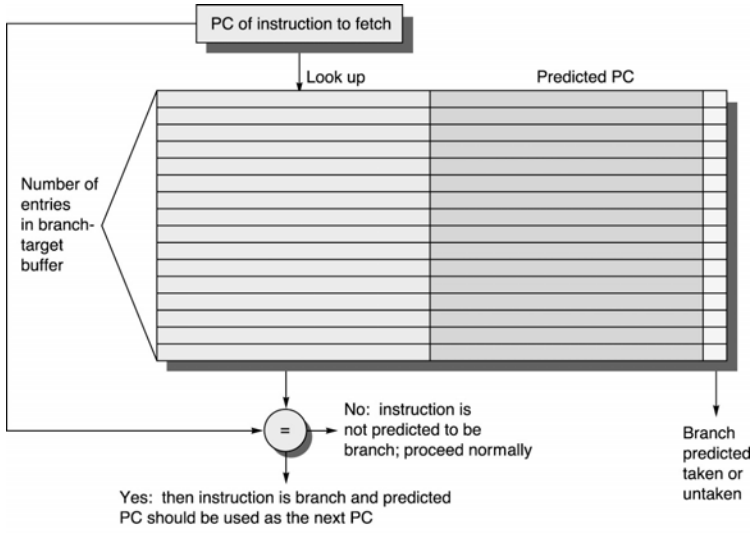
20



We earlier said that direction of branch is not known until after EX, so how come we are checking for branch direction in ID?

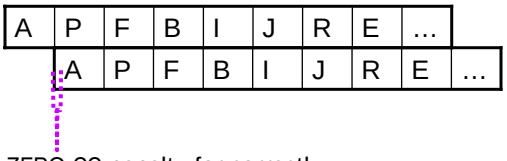
21

BHT + BTB



22

No Stall or Lost Work Cycle With BTB



ZERO CC penalty for correctly predicted branch with BTB

23