Reducing Branch Penalty
Branch penalty in dynamically scheduled processors: wasted cycles due to pipeline flushing on mis-predicted branches

Reduce branch penalty:
1. Predict branch/jump instructions AND branch direction (taken or not taken)
2. Predict branch/jump target address (for taken branches)
3. Speculatively execute instructions along the predicted path

Prediction and Prediction Output
Prediction is made for EVERY instruction
- The only ACCURATE input is the current PC
  - If pre-decoded, inst type is available
- Prediction is made on ALL types of instructions
- Prediction output is the next PC value (which is either current PC + 4 or a branch target)
- Three guesses are made: (1) if the next inst is a branch/jump at all; (2) if the "branch" would be taken; (3) what is the target of the "taken branch".

Mis-Prediction Cases
For predicted taken branches (fetch_pc != pc + 4), mis-predicted if the inst
- is not a branch/jump instruction; or
- target address was predicted wrong; or
- is a branch but not taken

For predicted not taken branches (fetch_pc == pc + 4), mis-predicted if the inst
- is a jump instruction; or
- is a branch instruction, AND the branch is taken
### Mis-prediction Detections and Feedbacks

**Detections:**
- At commit (most cases)
- At the end of decoding
  - The inst must be non-speculative

**Feedbacks:**
- From commit stage
- From decoding
- Or from WB if speculative feedback is allowed

---

### Branch (direction) Prediction

- Predict branch direction: taken or not taken (T/NT)

```plaintext
BNE R1, R2, L1
Not taken
L1: ...
```

- Static prediction: compilers decide the direction
- Dynamic prediction: hardware decides the direction using dynamic information
  1. 1-bit Branch-Prediction Buffer
  2. 2-bit Branch-Prediction Buffer
  3. Correlating Branch Prediction Buffer
  4. Tournament Branch Predictor
  5. and more ...

---

### Predictor for a Single Branch

**General Form**

1. Access
2. Predict Output T/NT
3. Feedback T/NT

**1-bit prediction**

- Predict Taken
- Predict Not Taken

---

### Branch History Table of 1-bit Predictor

**BHT also Called Branch Prediction Buffer in textbook**

- Can use only one 1-bit predictor, but accuracy is low
- BHT: use a table of simple predictors, indexed by bits from PC
- Similar to direct mapped cache
- More entries, more cost, but less conflicts, higher accuracy
- BHT can contain complex predictors
1-bit BHT Weakness

- Example: in a loop, 1-bit BHT will cause 2 mispredictions

- Consider a loop of 9 iterations before exit:
  ```
  for (i=0; i<9; i++)
    a[i] = a[i] * 2.0;
  ```
  - End of loop case, when it exits instead of looping again
  - First time through loop on next time through code, when it predicts exit instead of looping
  - Only 80% accuracy even if loop 90% of the time

2-bit Saturating Counter

- Solution: 2-bit scheme where change prediction only if get misprediction twice:

```
Predict Taken
11
T
01
Predict Not Taken
Predict Taken
10
NT
00
Predict Not Taken
```

Correlating Branches

**Code example showing the potential**

```java
If (d==0)
  d=1;
If (d==1)
  ...
```

**Assemble code**

- BNEZ R1, L1
- DADDIU R1,R0,#1
- L1: DADDIU R3,R1,#-1
- BNEZ R3, L2
- L2:

**Observation:** if BNEZ1 is not taken, then BNEZ2 is not taken

Correlating Branch Predictor

- Idea: taken/not taken of recently executed branches related to behavior of next branch (as well as the history of that branch behavior)

- Then behavior of recent branches selects between, say, 2 predictions of next branch, updating just that prediction

- (1,1) predictor: 1-bit global, 1-bit local

```
1-bits per branch
local predictors
```

```
1-bit global branch history
(0 = not taken)
```
ILP: Branch Prediction and Instruction Delivery

Branch target buffer, return address prediction, tournament predictor, high-performance instruction delivery

Correlating Branch Predictor

General form: (m, n) predictor
- m bits for global history, n bits for local history
- Records correlation between m+1 branches
- Simple implementation: global history can be stored in a shift register
- Example: (2,2) predictor, 2-bit global, 2-bit local

Branch Target Buffer

- Branch Target Buffer (BTB): Address of branch index to get prediction AND branch address (if taken)
  - Note: must check for branch match now, since can’t use wrong branch address
- Example: BTB combined with BHT

Accuracy of Different Schemes

4096 Entries 2-bit BHT
Unlimited Entries 2-bit BHT
1024 Entries (2,2) BHT

Extra prediction state bits
Yes: instruction is branch and use predicted PC as next PC
No: branch not predicted, proceed normally (Next PC = PC+4)
Return Addresses Prediction

- Register indirect branch hard to predict address
  - Many callers, one callee
  - Jump to multiple return addresses from a single address
    (no PC-target correlation)

- SPEC89 85% such branches for procedure return

- Since stack discipline for procedures, save return address in small buffer that acts like a stack: 8 to 16 entries has small miss rate

---

Tournament Branch Predictor

- Used in Alpha 21264: Track both “local” and global history
- Intended for mixed types of applications
- Global history: T/N/T history of past k branches, e.g. 0 1 0 1 0 1 (NT T NT T NT T)

---

Accuracy of Return Address Predictor

- SPEC89 85% such branches for procedure return
- Since stack discipline for procedures, save return address in small buffer that acts like a stack: 8 to 16 entries has small miss rate
Branch Prediction With n-way Issue
1. Branches will arrive up to $n$ times faster in an $n$-issue processor
2. Amdahl’s Law => relative impact of the control stalls will be larger in an $n$-issue processor

Integrated Instruction Fetch Units
1. Integrated branch prediction: branch predictor becomes part of the instruction fetch unit
2. Instruction prefetch: fetch ahead to deliver multiple instructions per cycle
3. Instruction memory access and buffering: may access multiple cache lines in one cycle, use prefetch to hide the cost
   • Another approach: trace cache

Instruction Fetch Unit
- Fetch predictor Predicts next fetch addresses to avoid fetch delay; may pre-predict branch direction; may be integrated with I-cache
- Branch predictor overrides and trains fetch predictor

Out-of-order Execution Engine

Pitfall: Sometimes bigger and dumber is better
- 21/64 uses tournament predictor (29 Kbits)
- Earlier 21/64 uses a simple 2-bit predictor with 2K entries (or a total of 4 Kbits)
- Branch predictor
- SPEC95 benchmarks, 21264 outperforms
  - 21264 avg. 11.5 mispredictions per 1000 instructions
  - 21164 avg. 16.5 mispredictions per 1000 instructions
- Reversed for transaction processing (TP)!
  - 21264 avg. 17 mispredictions per 1000 instructions
  - 21164 avg. 15 mispredictions per 1000 instructions
- TP code much larger & 21164 hold 2X branch predictions based on local behavior (2K vs. 1K local predictor in the 21264)
Dynamic Branch Prediction Summary

- Prediction becoming important part of scalar execution
- Branch History Table: the array of 2 bits FSMs
- Correlation: Recently executed branches correlated with next branch.
  - Either different branches (global history)
  - Or different executions of same branches (local history)
- Tournament Predictor: more resources to competitive solutions and pick between them
- Branch Target Buffer: include branch address & prediction
- Return address stack for prediction of indirect jump