CSE502 – Computer Architecture

Lecture 11
Instruction Level Parallelism (Chapter 3)
Branch Prediction

Dr. Iehul Yoon (ieyoon@sunykorea.ac.kr)

Slides adapted from:
Larry Wittie, SBU & John Kubiatowicz, EECS, UC, Berkeley

Outline
- ILP – Instruction Level Parallelism
- Compiler techniques to increase ILP
- Loop Unrolling
- Static Branch Prediction
- Dynamic Branch Prediction
- Overcoming Data Hazards with Dynamic Scheduling
  - (Start) Tomasulo Algorithm
  - Conclusion

Static (Compile-Time) Branch Prediction
- Prior lecture showed scheduling code around delayed branch
- To reorder code around branches, need to predict branch statically when compile
- Simplest scheme is to predict a branch as taken
- Average misprediction = untaken branch frequency = 34% in SPEC92 benchmarks

More accurate scheme predicts branches using profile information collected from earlier runs, and modify prediction based on last run.

Dynamic (Run-Time) Branch Prediction
- Why does prediction work?
  - Underlying algorithm has regularities
  - Data that is being operated on has regularities
  - Instruction sequences have redundancies that are artifacts of way that humans/compilers solve problems
- Is dynamic branch prediction better than static prediction?
  - Seems to be.
  - There are a small number of important branches in programs which have dynamic behavior
Dynamic (Run-Time) Branch Prediction

- Performance = \( f(\text{accuracy}, \text{cost of misprediction}) \)

- Branch History Table (or branch-prediction buffer)
  - Lower bits of PC address index table of 1-bit values
  - Effectively a cache where every access is a hit.
  - Says whether or not branch taken last time
  - No address check

- 1-bit BHT (1-bit predictor)
  - Remember last taken/not-taken per branch
  - Will cause two mispredictions per loop, (Average for loops is 9 iterations before exit):
    - End of loop case, when it exits instead of looping as before
    - First time through loop on next time through code, when it predicts exit instead of looping

Branch History Table (BHT) Accuracy

- Mispredict because either:
  - Make wrong guess for that branch
  - Got branch history of wrong branch when index the table (same low address bits used for index).

- 4096 entry BH table (2-bit prediction buffer):

Correlated Branch Prediction

- Idea: Record \( m \) most recently executed branches as taken or not taken, and use that pattern to select the proper \( n \)-bit branch history table

- In general, \((m,n)\) predictor means record last \( m \) global branches to select between \( 2^m \) local branch history tables, each with \( n \)-bit counters
  - Thus, the old 2-bit BHT is a \((0,2)\) predictor
  - Global Branch History: \( m \)-bit shift register keeping Taken/Not_Taken status of last \( m \) branches anywhere.
  - Each entry (row for given set of address bits) in table has \( 2^m \) \( n \)-bit predictors.

```c
if (aa == 2) { 
    aa = 0; 
    if (bb == 2) { 
        bb = 0; 
        if (aa != bb) { 
            // do something 
        }
    }
}
```
Correlating Branch Predictors

A (2,2) predictor with 16 sets of four 2-bit predictors
- Behavior of most recent 2 branches selects between four predictions for next branch, updating just that prediction

Branch address selects row of entries

4 address bits

2-bits per branch predictor

2-bit global branch history

Or, 4 addr bits + 2 history bits give us 6-bit index into $2^6 = 64$ predictors, each having two bits $\Rightarrow 128$ total bits.

Accuracy of Different Schemes

Tournament Predictors

- Multilevel branch predictor
- Use $n$-bit saturating counter to choose between predictors
  - Usual choice is between global and local predictors

Tournament Predictors for Alpha 21264

- Selector
  - 4K 2-bit counters indexed by local branch address.
- Global predictor
  - 4K entries index by history of last 12 branches ($2^{12} = 4K$)
  - Store history in a global history register
  - Each entry is a standard 2-bit predictor
  - Irrelevant to specific branch address
- Local predictor
  - Top-level: local history table: 1K 10-bit history of (local) branch outcomes, indexed by branch address – 10 previous branch outcomes per branch
  - The 10-bit pattern is used to index a 1K table with 3-bit saturating counters
- Predictor Size
  - $4K*2 + 4K*2 + 1K*10 + 1K*3 = 29K$
  - In 2005, tournament predictor using about 30K were standard (e.g., Pentium 4 and Power 5)
Comparing Predictors (Fig. 2.8)

- Advantage: tournament predictor can select the right predictor for a particular branch
  - Particularly crucial for integer benchmarks.
  - A typical tournament predictor will select the global predictor almost 40% of the time for the SPEC Integer benchmarks and less than 15% of the time for the SPEC FP benchmarks.

![Graph showing comparison of predictors]

Other examples of Branch Predictors

- Pentium
  - 2-bit local predictor
    - Direct jump from (00) to (11) state
- Pentium MMX, Pentium Pro, Celeron, Pentium II
  - (4, 2) correlating predictor
- Pentium III
  - 2-level adaptive tournament predictor
  - 512-entry Branch Target Buffer (BTB) in next slides
- Pentium IV
  - 4096-entry BTB and execution trace cache
- Intel Core i7
  - Two-level predictor (smaller first-level, bigger second-level predictor)
  - Each combines three predictors
    - (1) Simple 2-bit predictor, (2) global history predictor, (3) a loop exit predictor

Pentium 4 Misprediction Rate
(per 1000 instructions, not per branch)

- ~6% misprediction rate per branch SPECint
  - (19% of INT instructions are branch)
- ~2% misprediction rate per branch SPECfp
  - (5% of FP instructions are branch)

![Bar chart showing misprediction rates]

Floor Plan of Pentium Processor

![Diagram of the Pentium processor floor plan]
Branch Target Buffers (BTB)

- Branch target calculation is costly and stalls the instruction fetch one or more cycles.
- BTB stores branch PCs and target PCs the same way that caches store addresses and data blocks.
- The PC of a branch is sent to the BTB.
- When a match is found, the corresponding predicted target PC is returned.
- If the branch is predicted to be Taken, instruction fetch continues at the returned predicted PC.
- Extra logic verifies predicted PC was correct andflushes all incorrectly fetched instructions from pipeline(s) without storing any results.

Dynamic Branch Prediction Summary

- Prediction becoming important part of execution.
- Branch History Table: 2 bits for loop accuracy.
- Correlation: Recently executed branches correlated with next branch.
  - Either different branches (GA)
  - Or different executions of same branches (PA).
- Tournament predictors take insight to next level, by using multiple predictors.
  - Usually one based on global information and one based on local information, and combining them with a selector.
  - In 2006, tournament predictors using ~30K bits are in processors like the Power5 and Pentium 4.
- Branch Target Buffer: include branch address & prediction.