Branch Prediction

vs

Execution Time Prediction

Jakob Engblom, PhD

Uppsala University & Virtutech Inc.

jakob.engblom@it.uu.se
jakob@virtutech.com
The Question

★ Branch prediction
  ◆ Performance enhancing technique
  ◆ Necessary with deep pipelines
  ◆ Works well, on average

★ Execution time prediction
  ◆Determining the extremes
  ◆ Especially the worst case

★ Does “BP” make “ETP” harder?
Execution Time Estimates

**WCET** = Worst case (main interest here)

**BCET** = Best case

**ACET** = Average case

- **WCET** = Worst case
- **BCET** = Best case
- **ACET** = Average case
Branch Prediction
The Performance Problem

```
cmp r7,5
bne B
A:
  add r4,r5
  ...
B:
  bset r5,1
```

Conditional branch: execution will continue at A or B

You need to redirect instruction fetch here

Result of branch is not known until here

= wait to see where branch goes = stall
Static Techniques

- Keep fetching ahead
  - Always assume not taken
  - Or introduce "branch delay slot"

- BTFN
  - Backwards-taken
  - Forwards-not taken
  - Recognize branches in IF or ID
  - Make speculative decision early
  - About 70% correct

NEC V850, ARM7

original
SPARC & MIPS

ARM10, + base case in more advanced predictors
”History will repeat itself”
- Use history of taken/not taken

One-level:
- One counter per branch
  - Actually a state machine
  - Usually with hysteresis

Implementation:
- Cache of counters
- Indexed by branch address

Pentium 1, Alpha 21064, UltraSparc II
”History has a pattern”
- Use *pattern* of taken/not taken
  - ”taken every other time”, for example
- History register tracks outcomes
  - History per branch or global
- Table of counters
  - Combination of history and address
  - 2D table, XOR, ... lots of possibilities

UltraSparc III, Athlon, Pentium 3, Pentium 4, PowerPC G3, G4

| history: 01001... |
| address: 11100... |
The Experimental setup
Experimental Setup

```
for{k=1; k<32; k++)
{
    starttimer();
    for(n=0; n < 10000000; n++)
    {
        for(i=0; i < k; i++)
        {
            __nop();
        }
    }
    stoptimer();
    recordtime();
}
```
Baseline Result

**Static prediction: total time**

![Graph showing V850E Time with a monotone increase and a smooth straight line. The graph indicates perfectly easy to predict.]
Baseline Result

**Static prediction: / inner count**

Smooth monotone decrease

Cost of outer loop is spread across more & more iterations of inner loop
The Results
One-Level Dynamic

UltraSparc II Time

Takes some time for predictor to tune in

Monotone increase, but not exactly smooth

Analyzing or measuring max # iterations is safe
Two-Level Dynamic, Local

**Inversion:**
doing more iterations takes less time

Increases the search space for the worst case considerably
Inversions Explained

Cost of the mispredict is greater than the cost of executing an extra inner loop

Cost of the mispredict is greater than the cost of executing an extra inner loop

n+1 iterations, takes T cycles

n iterations, takes \( T \) cycles
Two-Level Dynamic, Local

Wide variation in time per iteration

Four iterations, length of history register, is a local minimum
Two-Level Dynamic, Global

Generalizing from early trend gives a too low execution time estimate.

Nice smooth curve, with a single bump.

But still inversions.

Looks better, but no easier than the local two-level predictor.

Maybe inversions stop appearing.
Two-Level Dynamic, Global

UltraSparc III Time/Count

Drop here is effect of BTFN initial guess

Bump after the history register wraps around

This is the lowest point on the curve!
Pentium 4

Very uneven curve, strangest machine in all experiments

Inversions keep occurring

Very difficult to predict the worst case
Pentium 4

BTFN assumption is false for 1 iteration

Very impressive average-case performance

Worse after 16-17 iterations!
Sneak peak at future work
Testing Size of Inner Loop

- Pack more NOPs into the inner loop
- Curve gets smoother as size of inner loop increases
- Inversions disappear with sufficiently large loop body?
- Larger loop bodies can hide BP effects
Conclusions
Conclusions

» Static branch prediction
  ◆ No problem for WCET estimation

» Dynamic branch prediction
  ◆ Inversions: more work = less time
  ◆ Execution time increases unevenly
  ◆ Not easy to find the worst case

» Overall: BP makes ETS harder
  ◆ To minimize impact, maximize size of loop bodies
The End!

Questions?