

ECE 571 – Advanced Microprocessor-Based Design Lecture 11

Vince Weaver

`http://web.eece.maine.edu/~vweaver`

`vincent.weaver@maine.edu`

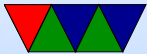
25 September 2020

Announcements

- HW#4 will be posted
- Last lecture had the for/while branch counts slightly wrong, off-by-one. Both of those check condition at entry so have extra branch (100 not 99)



How can we Improve Things?

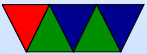


Branch Prediction Hints

- Give compiler (or assembler) hints
- `likely()` (maps to `__builtin_expect()`)
- `unlikely()`
- on some processors, (p4) hint for static
- others, just move unlikely blocks out of way for better L1\$ performance
- Linux did this – but turns out people can be bad at hinting



Dynamic Branch Prediction

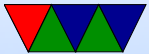
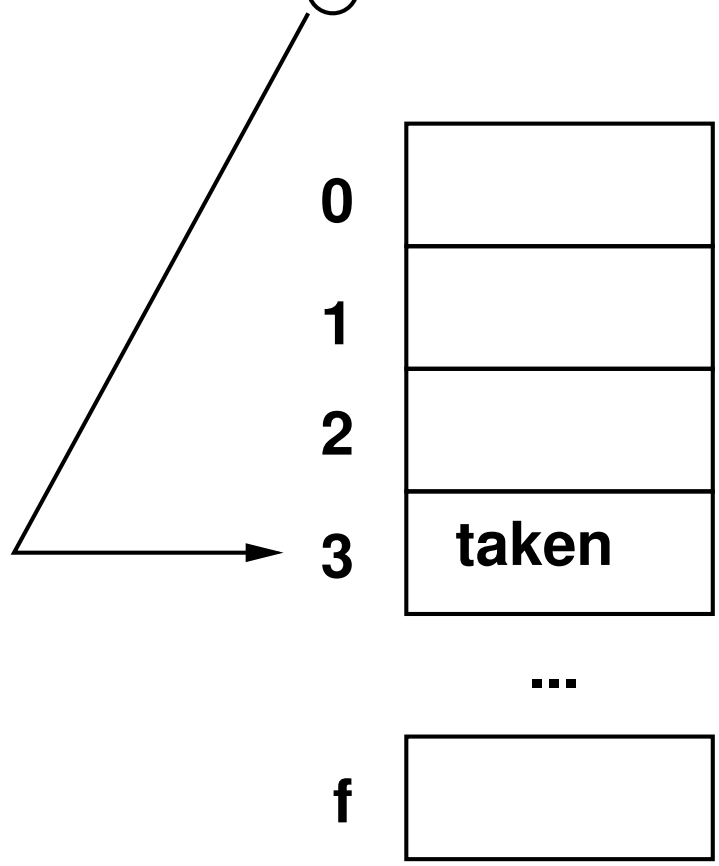


One-bit Branch History Table

- Table, likely indexed by lower bits of instruction (Why low bits and not high bits?)
- Can have more advanced indexing to avoid aliasing no-tag bits, unlike caches aliasing does not affect correct program execution
- One-bit indicating what the branch did last time
- Update when a branch miss happens



0x1000 0003 : bne PC+45



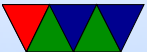
Branch History Table Behavior

- Two misses for each time through loop. Wrong at exit of loop, then wrong again when restarts. (so actually worse than static on loops)
- If/Then potentially better than static if long runs of true/false but can be worse if completely random.



Aliasing

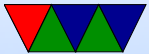
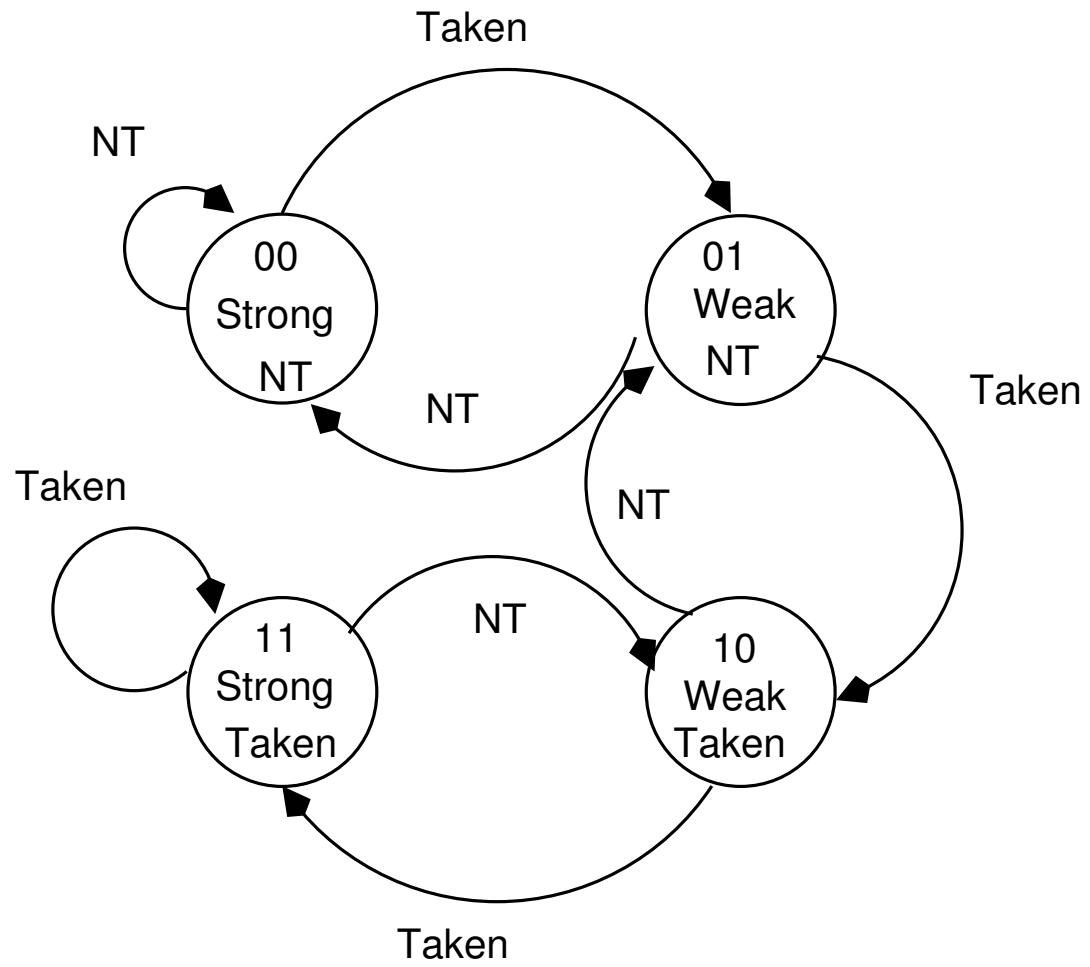
- Is it bad? Good?
- Does the O/S need to save on context switch?
- Do you need to save if entering low-power state?



Two-bit saturating counter

- Hysteresis
- Use saturating 2-bit counter
- If 3/2, predict taken, if 1,0 not-taken. Takes two misses or hits to switch from one extreme to the next, letting loops take only one mispredict.
- Needs to be updated on every branch, not just for a mispredict



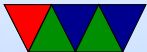
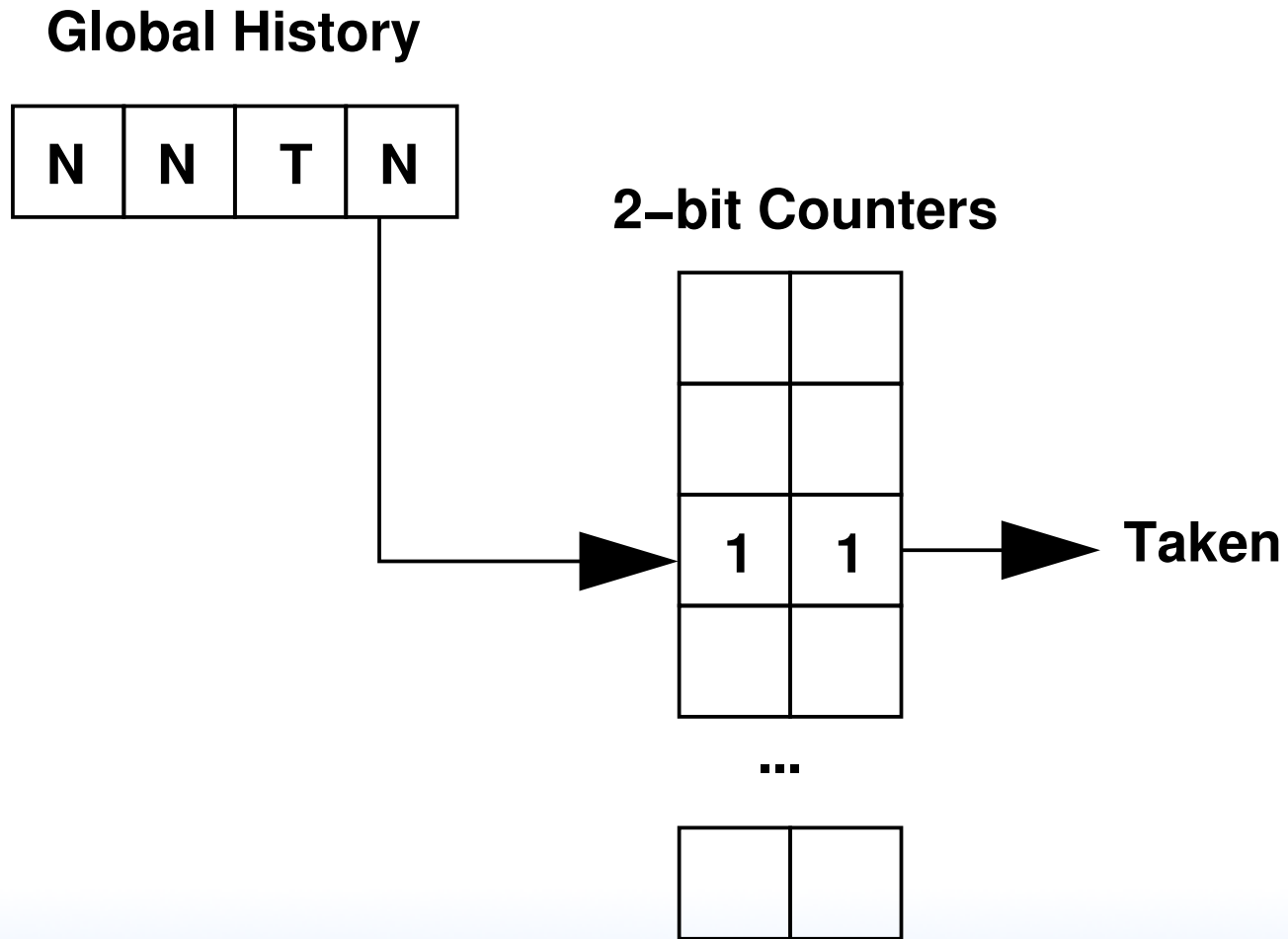


History

- Can use branch history as index into tables
- Use a shift register to hold history
- Global vs Local
 - Global: history is all branches
 - Local: store branch history on a branch by branch basis

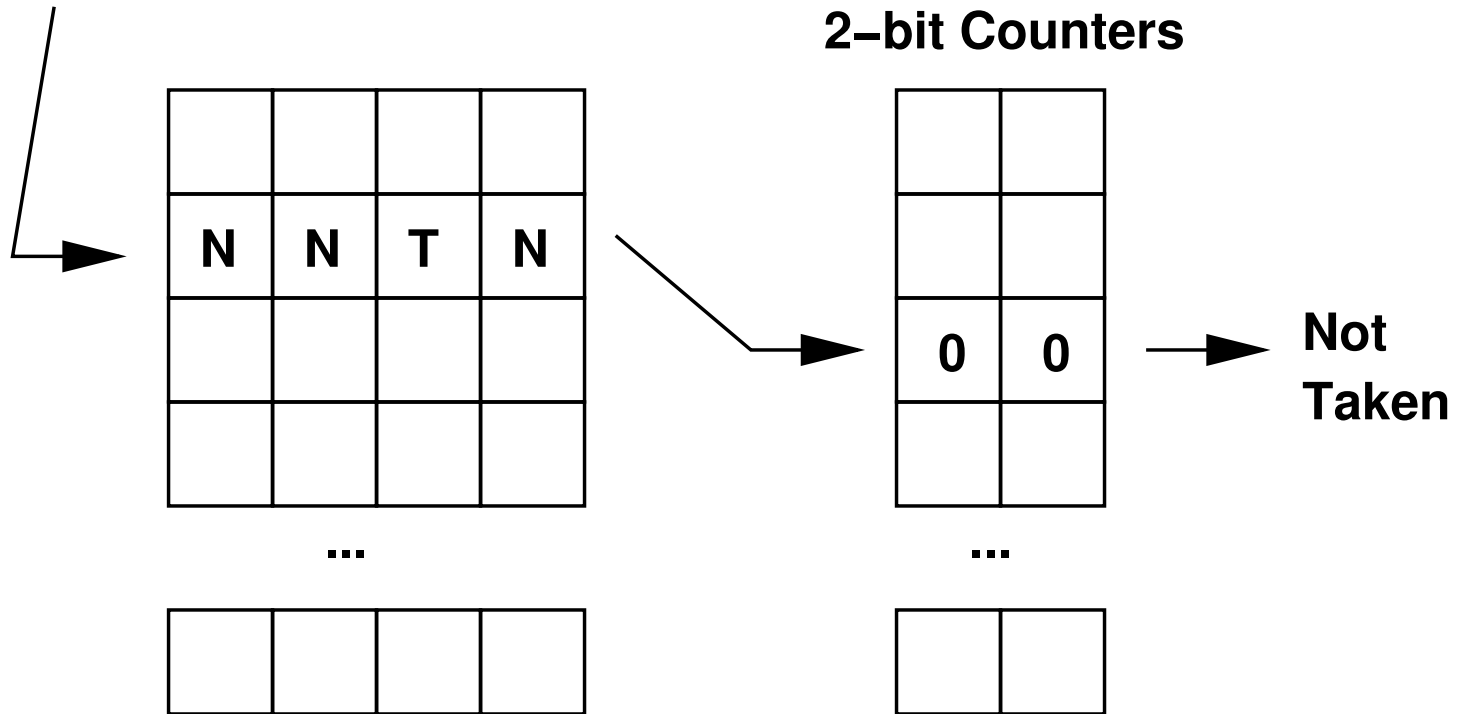


Global Predictor



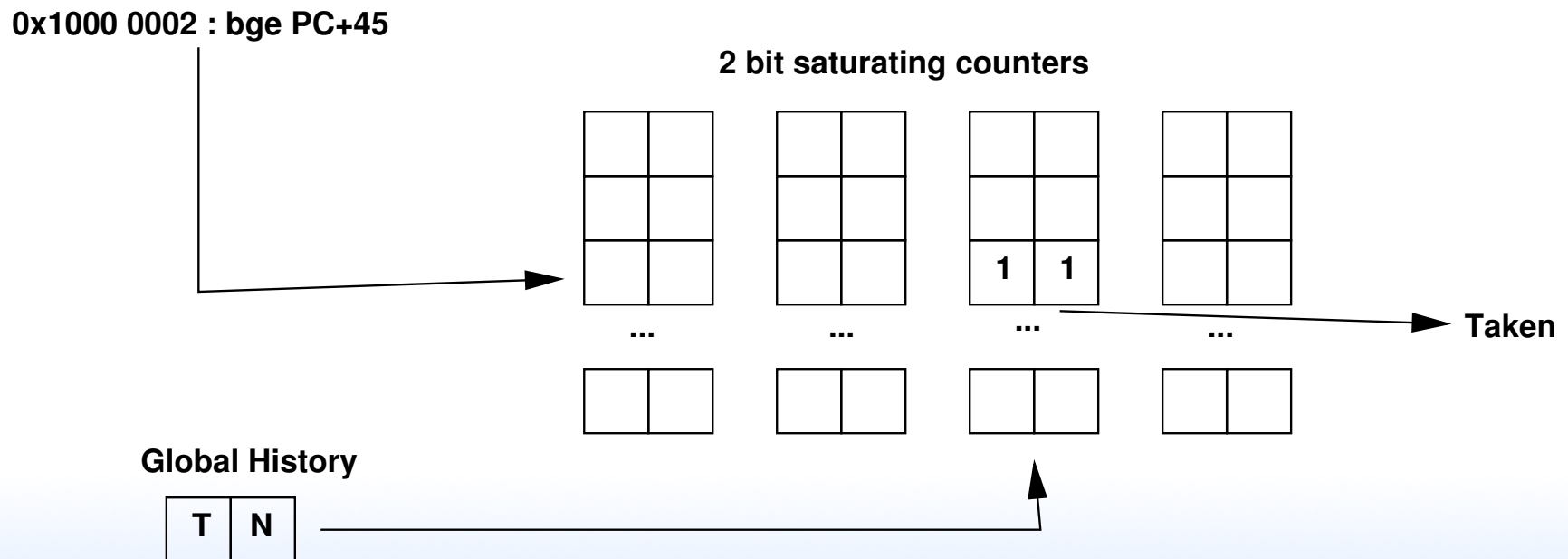
Local Predictor

0x8000 0001 : bne PC+45



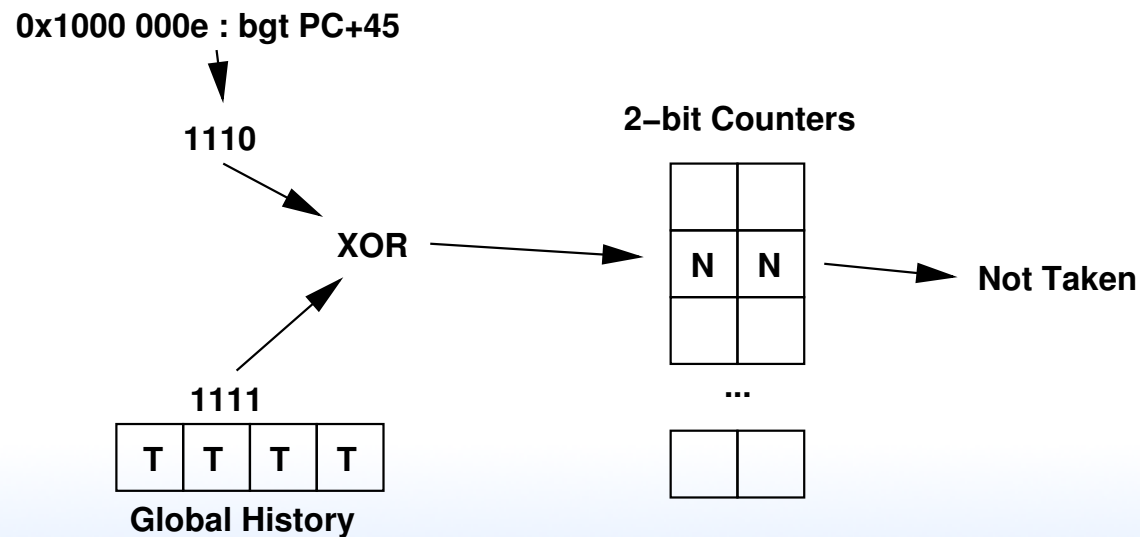
Correlating / Two Level Predictors

- Take history into account.
Break branch prediction for a branch out into multiple locations based on history.



gshare

- Xors the global history with the address bits to get which line to use.
- Benefits of 2-level without the extra circuitry



Tournament Predictors

- Which to use? Local or global?
- Have both. How to know which one to use? Predict it!
- 2-bit counter remembers which was best.



Perceptron

- There are actually Branch Prediction Competitions
- The winner the past few times has been a “Perceptron” predictor
- Neural Networks
- Samsung M1 processor at Hotchips’16 says it has neural-net branch predictor
- AMD Zen uses hashed perceptron



Comparing Predictors

- Branch miss rate not enough
- Usually the total number of bits needed is factored in
- May also need to keep track of logic needed if it is complex.
- Speed: what if it takes 20 cycles to do a prediction?



Branch Target Buffer

- Instruction fetcher needs to know next instruction to fetch even before instructions decoded
- Predicts the actual destination of addresses.
- Indexed by whole PC. May be looking up before even know it is a branch instruction.
- Only need to store predicted-taken branches. (Why? Because not-taken fall through as per normal).



Return Address Stack

- Function calls can confuse BTB. Multiple locations branching to same spot. Which return address should be predicted?
- Keep a stack of return addresses for function calls
- Playing games with size optimization and fallthrough/tail optimization can confuse.



Adjusting Predictor on the Fly

Some processors let you configure predictor at runtime.

MIPS R12000 let you

ARM possibly does.

Why is this useful?

In theory if you have a known workload you can pick the one that works best.

Also if realtime you want something that is deterministic, like static prediction.

Also Good for simulator validation



Cortex A9 Branch Predictor

From the Manual:

- two-level prediction mechanism, comprising: a two-way BTAC of 512 entries organized as two-way x 256 entries
- a Global History Buffer (GHB) with 4096 2-bit predictors
- a return stack with eight 32-bit entries.
- It is also capable of predicting state changes from ARM to Thumb, and from Thumb to ARM.



Core2 i7 Branch Predictor

From H&P:

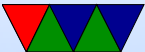
- Small 1 prediction/cycle predictor
- Larger (but slower) backup predictor that can correct the first if it's wrong
- Tournament predictor made up of
 - 2-bit
 - Global history
 - Loop-exit



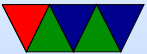
Example

Code in perf_event validation tests for generic events.

http://web.eece.maine.edu/~vweaver/projects/perf_events/validation/



Example Results



Part 1

Testing a loop with 1500000 branches (100 times):

On a simple loop like this, miss rate should be very small.

Adjusting domain to 0,0,0 for ARM

Average number of branch misses: 685

Part 2

Adjusting domain to 0,0,0 for ARM

Testing a function that branches based on a random number

The loop has 7710798 branches.

500000 are random branches; 250699 of those were taken

Adjusting domain to 0,0,0 for ARM

Out of 7710798 branches, 291081 were mispredicted

Assuming a good random number generator and no freaky luck

The mispredicts should be roughly between 125000 and 375000

Testing ‘‘branch-misses’’ generalized event...

PASSED



Value Prediction

- Can we use this mechanism to help other performance issues?
What about caches?
- Can we predict values loaded from memory?
- Load Value Prediction. You can, sometimes with reasonable success, but apparently not worth trouble as no vendors have ever implemented it.

