Branch Prediction using Pre-Computed Branches

Andy Meneely

Calvin College

April 10, 2006
Branch Prediction

- 1 in 7 assembly instructions are branches
- Being able to predict branches is a major concern in architecture
- Most traditional schemes have used history
  - Good, but not fully accurate
  - After many encounters, a predictor could be wrong.
BTC: Branch Target Cache

- Traditional way of branch prediction
- Keep a Prediction Table (PT)
  - Keep the instruction’s PC address
  - Use a bit for if it was taken last time or not
- Fast, but not fully accurate
PCB: Pre-Computed Branches

- Keep track of every branch and its operands in the PT
- Use the register’s value to predict the behavior of the branch
- The first time a branch is encountered, a miss might occur
- Perfect prediction once the branch behavior is stored
Many additions to the PT and RU for PCB.

- **Changes to the PT**
  - PC1,PC2 - PC pointers to the branch’s operand producers
  - OPC - the opcode of the branch (BNE, BEQ, etc...) 
  - nOP1 and nOP2 - register names of branch operands 
  - PRED - the prediction of the outcome 

- **Changes to the Register Unit**
  - Keep the same register file with the data value
  - LDPC - most recent instruction that wrote in that register
  - RC - Reference Counter. Incremented when register is used in a branch, decremented when a branch is evicted.
Pseudo-Pseudocode for the PCB Algorithm...
Is PCB Any Good? Well...

- **Pros**
  - Great if there’s a ton of branches
  - Outperforms all other branch prediction schemes in terms of prediction branches
  - Introduces an interesting idea to the field

- **Cons**
  - Adds a lot of space to the PT and RU
  - May speed up branches, but adds overhead to every non-branch instruction

- **Overall Complexity**
  - PCB: 105 KBits Complexity
  - BTC: 34 KBits Complexity
Any Questions?