

CMSC 611: Advanced Computer Architecture

Branch Prediction

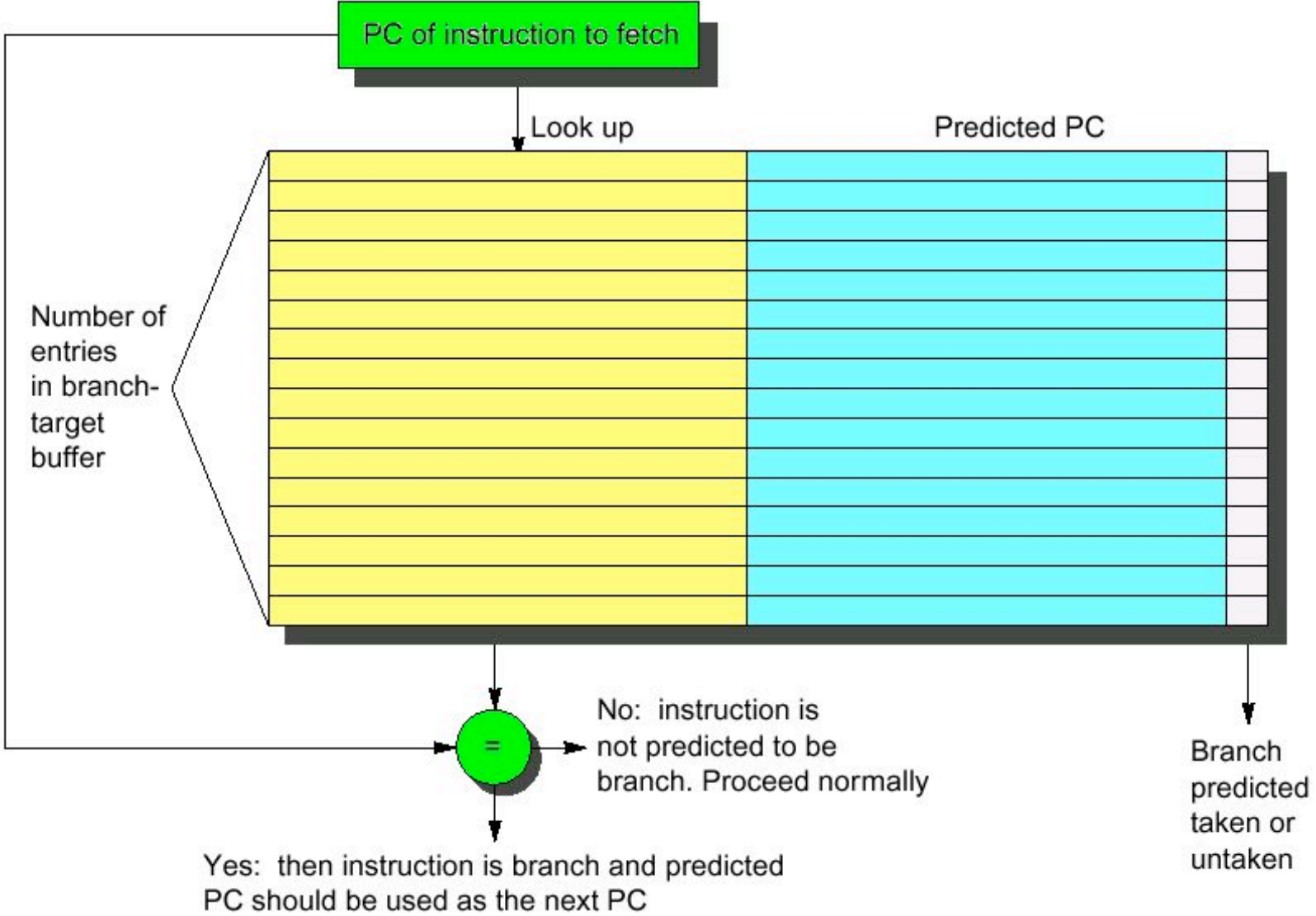
Branching Dilemma

- With the increased pipeline throughput, control dependence rapidly becomes the limiting factor to the amount of ILP
- For pipelines that issue n -instructions per clock cycle, the negative impact of stalls caused by control hazards magnifies
- Compiler-based techniques rely on static program properties to handle control hazards
- Hardware-based techniques refer to the dynamic behavior of the program to predict the outcome of a branch

Branch Target Cache

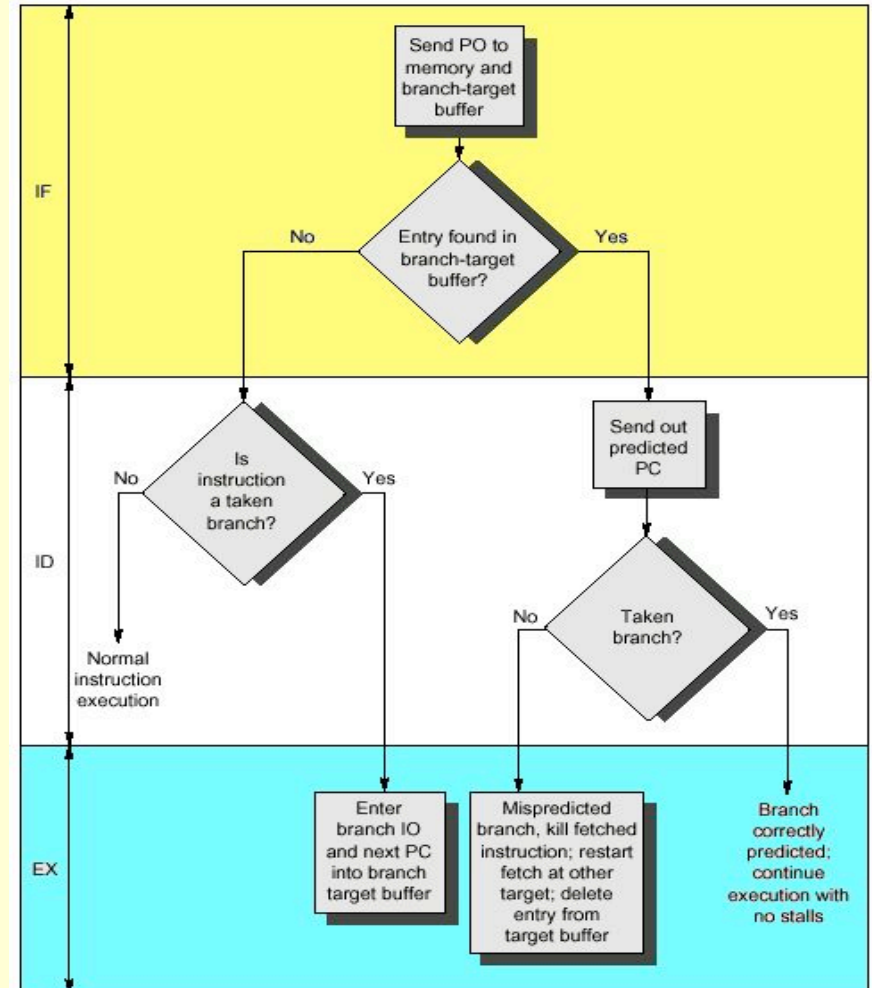
- Predict not-taken: still stalls to wait for branch target computation
- If address could be guessed, the branch penalty becomes zero
- Cache predicted address based on branch address
- Complications for complex predictors: do we know in time?

Branch Target Cache



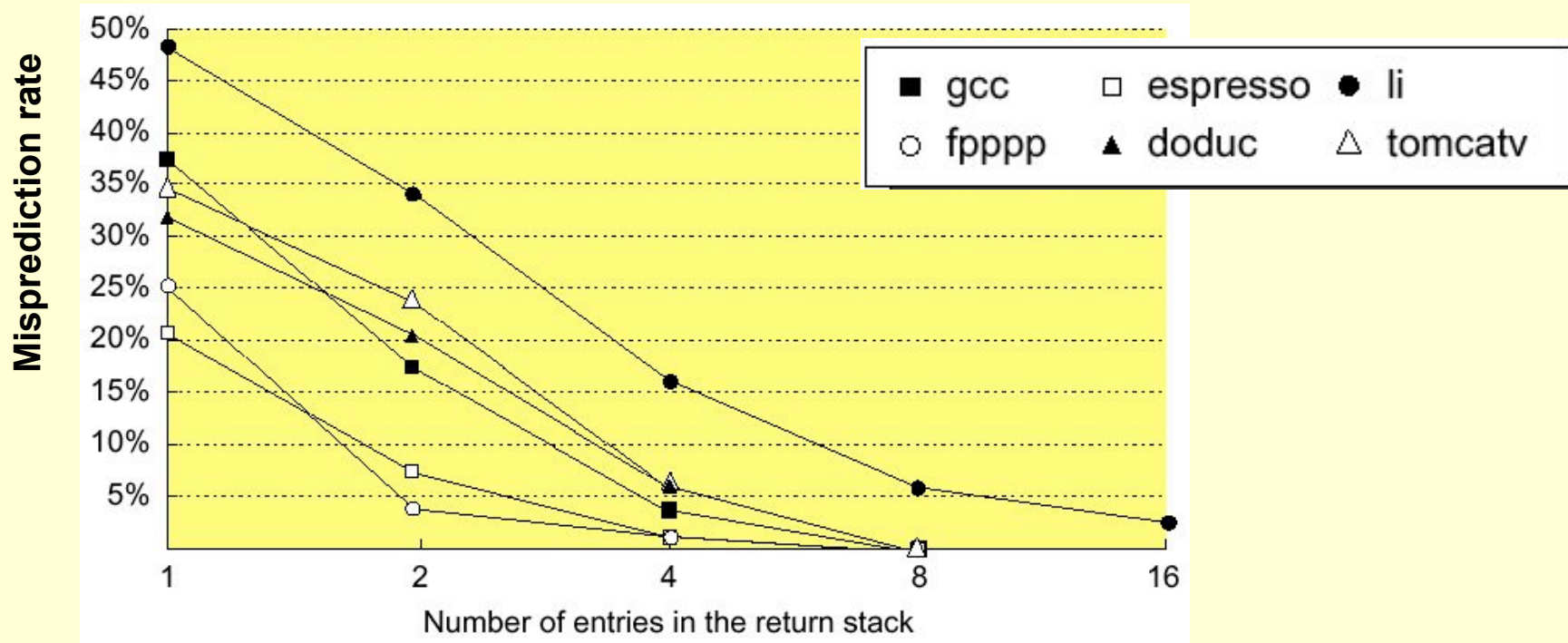
Handling Branch Target Cache

- No branch delay if the a branch prediction entry is found and is correct
- A penalty of two cycle is imposed for a wrong prediction or a cache miss
- Cache update on misprediction and misses can extend the time penalty
- Dealing with misses or misprediction is expensive and should be optimized



Return Address Cache

- Branch target caching can be applied to expedite unconditional jumps (branch folding) and returns for procedure calls
- For calls from multiple sites, not clustered in time, a stack implementation of the branch target cache can be useful

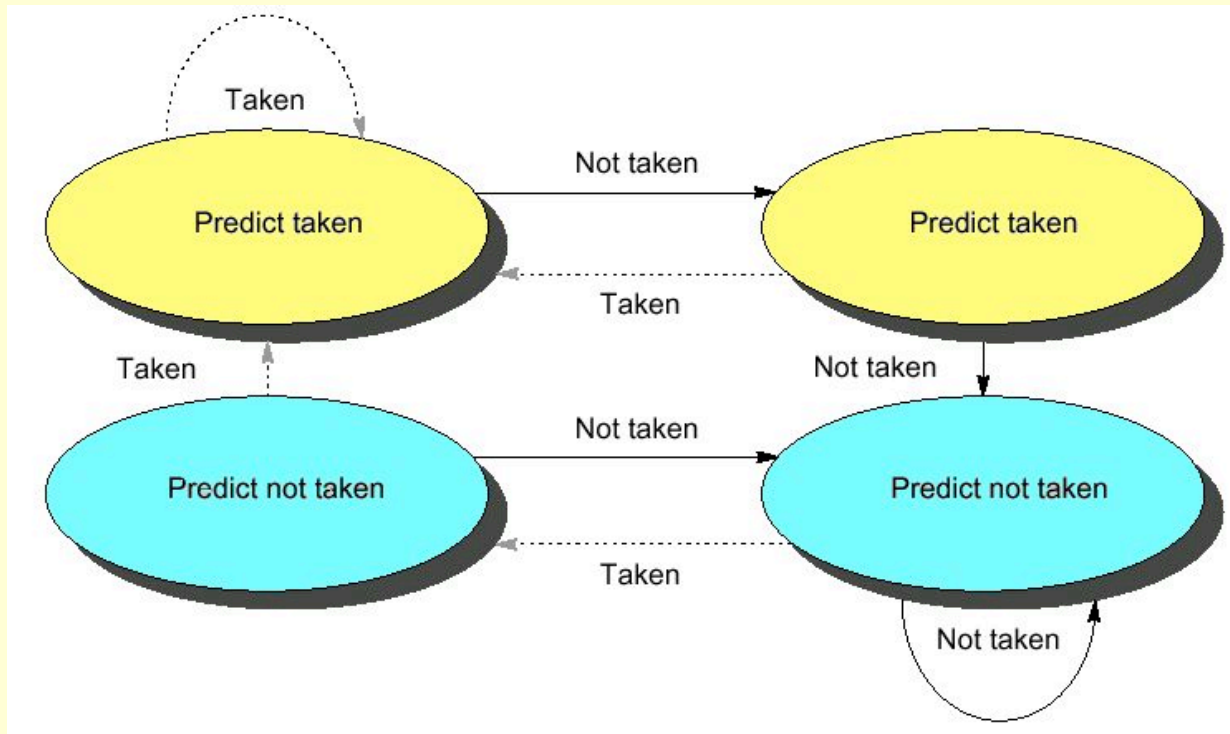


Basic Branch Prediction

- Simplest dynamic branch-prediction scheme
 - use a branch history table to track when the branch was taken and not taken
 - Branch history table is a small 1-bit buffer indexed by lower bits of PC address with the bit is set to reflect the whether or not branch taken last time
- Performance = $f(\text{accuracy, cost of misprediction})$
- Problem: in a nested loop, 1-bit branch history table will cause two mispredictions:
 - End of loop case, when it exits instead of looping
 - First time through loop on next time through code, when it predicts exit instead of looping

2-bit Branch History Table

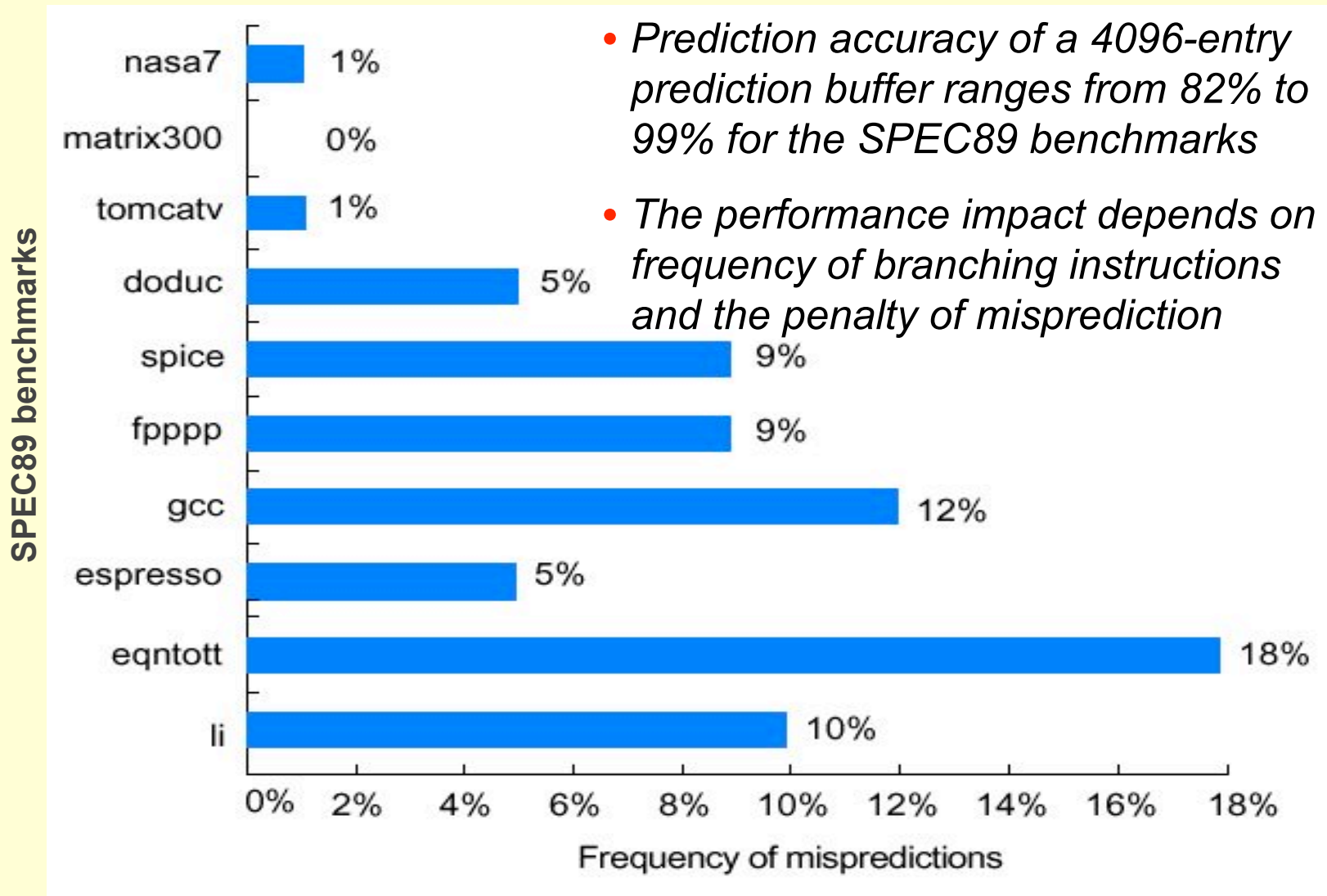
- A two-bit buffer better captures the history of the branch instruction
- A prediction must miss twice to change



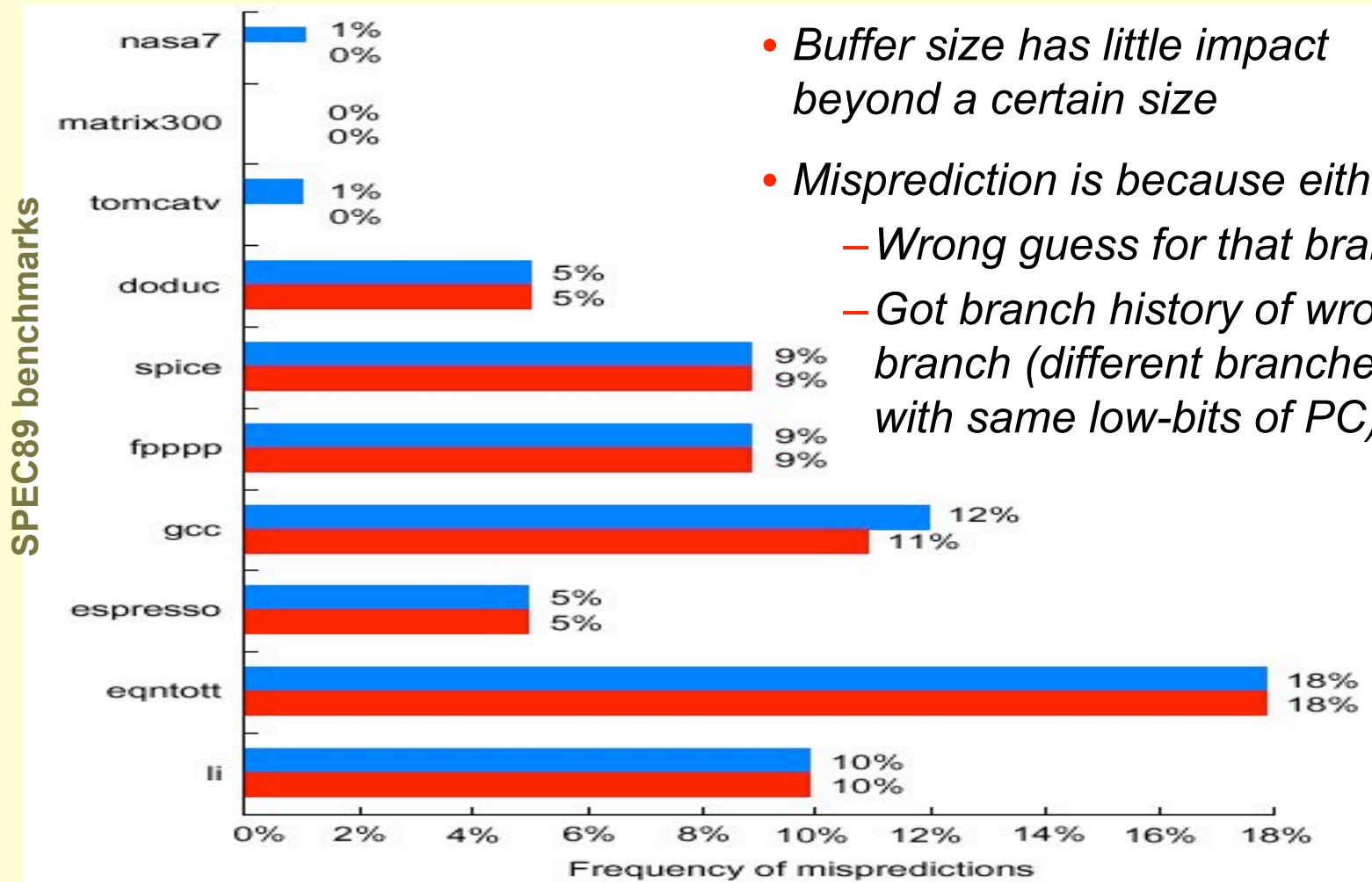
N-bit Predictors

- 2-bit is a special case of n-bit counter
 - For every entry in the prediction buffer
 - Increment/decrement if branch taken/not
 - If the counter value is one half of the maximum value ($2^n - 1$), predict taken
- Slow to change prediction, but can change

Performance of 2-bit Branch Buffer



Optimal Size for 2-bit Branch Buffers



- Buffer size has little impact beyond a certain size
- Misprediction is because either:
 - Wrong guess for that branch
 - Got branch history of wrong branch (different branches with same low-bits of PC)

■ 4096 entries (2 bits/entry)

■ Unlimited entries (2 bits/entry)