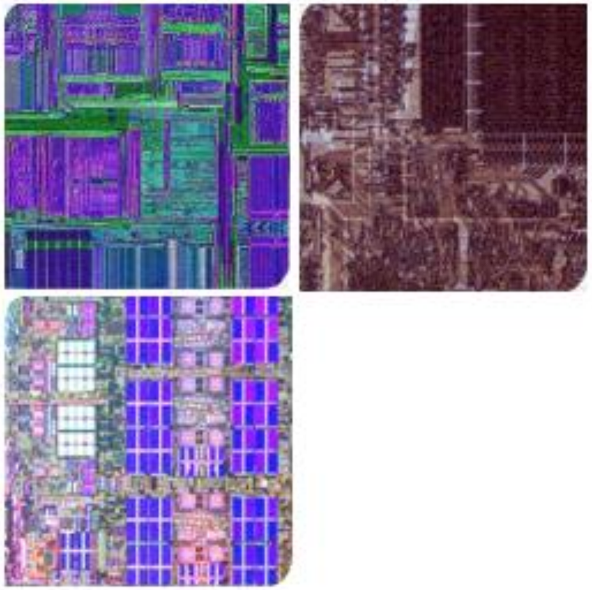
Branch Prediction



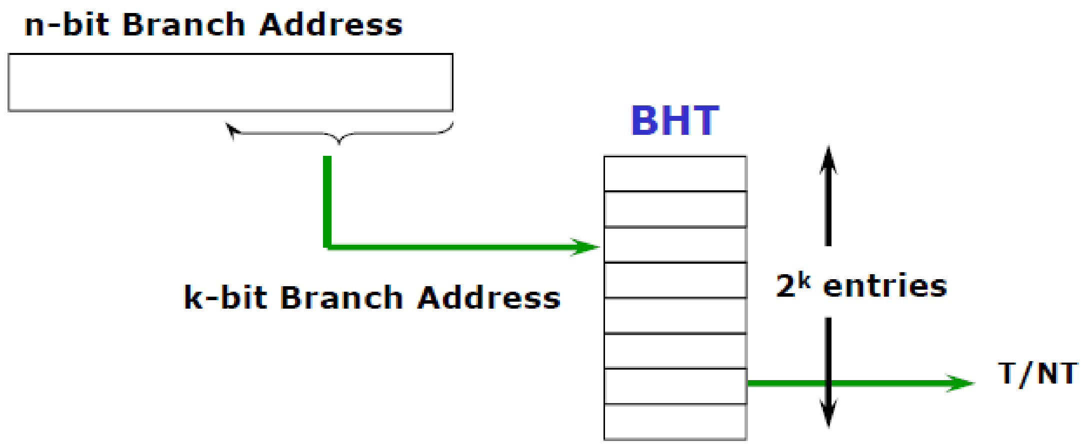
Branch Prediction

- Static branch prediction
  - Branch can be highly predicted at compile time
  - Branch predicted as taken
  - Misprediction rate for SPEC programs is 34%
  - Use profile information collected from earlier runs
  - Misprediction rate is higher for integer programs

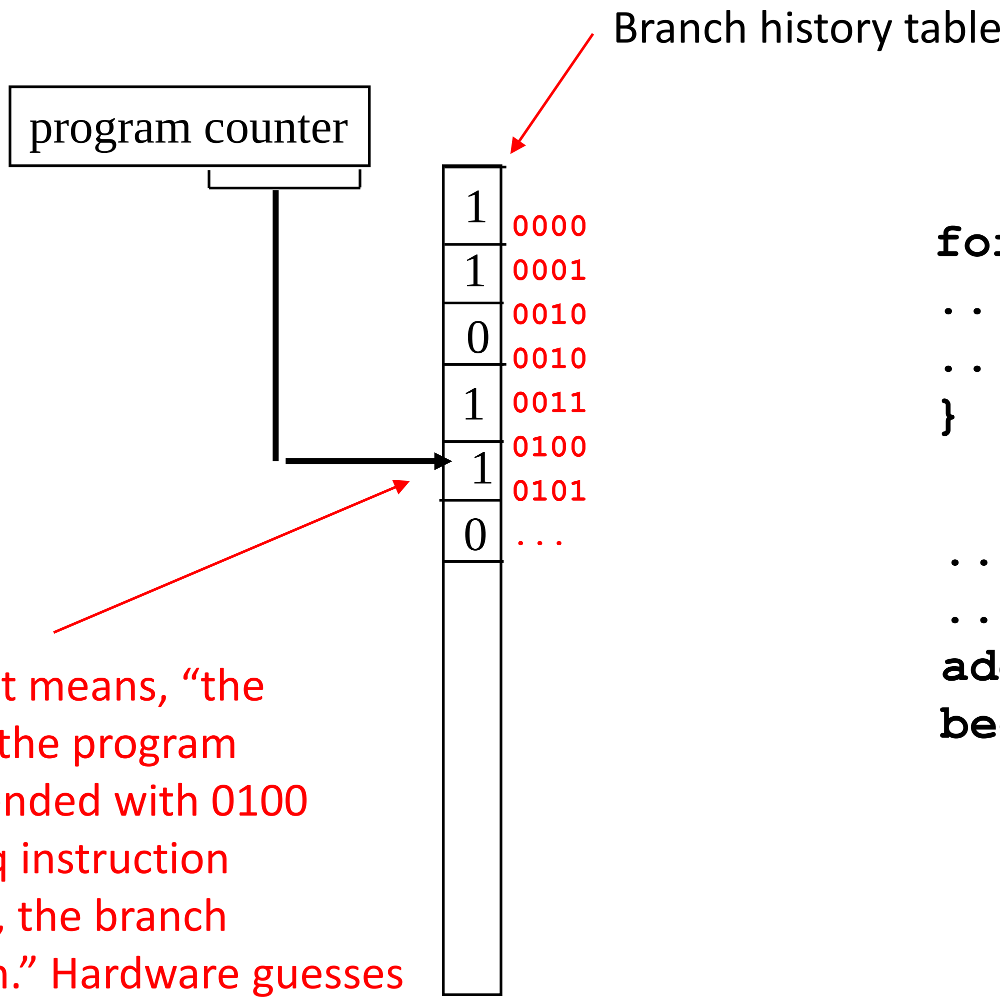
- Dynamic branch prediction
  - Learn branch behavior autonomously
  - Branch prediction buffer or branch history table

Dynamic Branch Prediction

- Branch prediction buffer – a small memory indexed by the lower portion of the address of the branch instruction
- Memory contains a bit that says whether the branch was taken recently or not
- 1- bit prediction scheme
  - 0 - not taken, 1- taken
  - Prediction is a hint that is assumed to be correct and fetching begins in the predicted direction
  - If the hint turns out to be wrong, the prediction bit is inverted and stored back



Branch Prediction



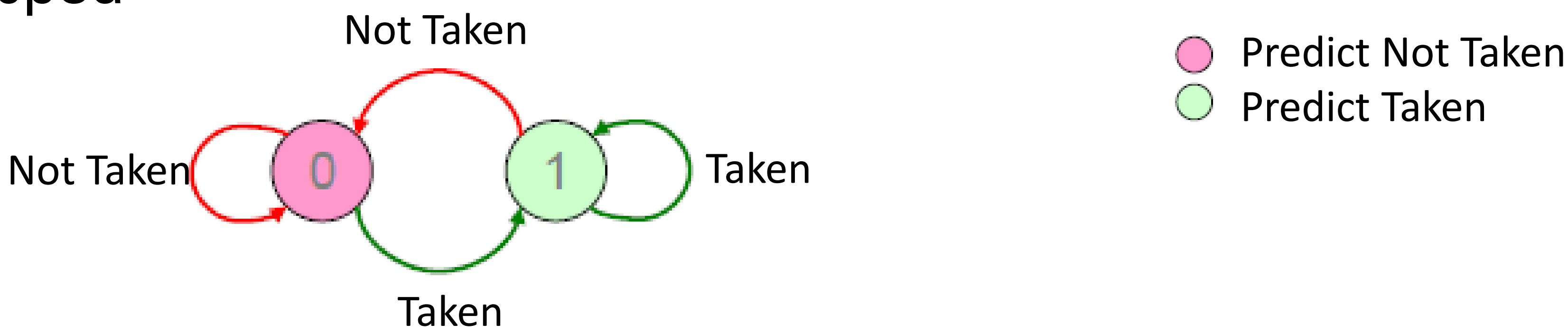

```
for (i=0;i<10;i++) {
...
...
}
```

```
...
...
add  $i, $i, #1
beq $i, #10, loop
```

This ‘1’ bit means, “the last time the program counter ended with 0100 and a beq instruction was seen, the branch was taken.” Hardware guesses it will be taken again.

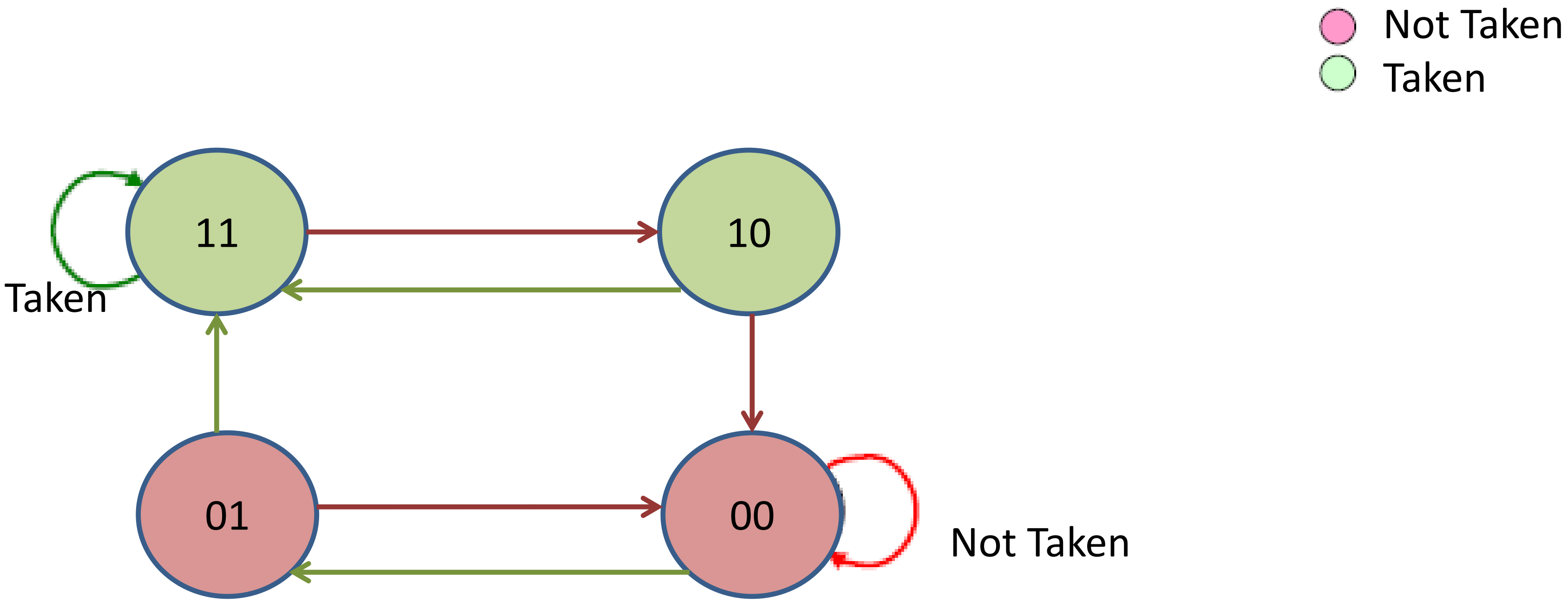
1- bit Prediction

- Even if a branch is almost always taken, it is likely to predict incorrectly twice, rather than once, when it is not taken, since the misprediction causes the prediction bit to be flipped



2- bit Prediction

- To overcome this, 2 bit prediction schemes are used. In this prediction scheme a prediction must miss twice before it is changed.



Correlating Branch Predictors

- Branch predictors that use the behavior of other branches to make a prediction are called correlating predictors or two-level predictors.
- For example, a (1,2) predictor uses the behavior of the last branch to choose from among a pair of 2-bit branch predictors in predicting a particular branch.
- In the general case, an *(m,n)* predictor uses the behavior of the last *m* branches to choose from *2m* branch predictors, each of which is an n-bit predictor for a single branch.
- The global history of the most recent *m* branches can be recorded in an m-bit shift register, where each bit records whether the branch was taken or not taken.
- The branch-prediction buffer can then be indexed using a concatenation of the low order bits from the branch address with the m-bit global history.

Correlating Branch Predictors

- The number of bits in an *(m,n)* predictor is

  $2m \times n \times$ Number of prediction entries selected by branch address

- Example: How many bits are in the (0,2) branch predictor with 4K entries?

  The predictor with 4K entries has

  $2^0 \times 2 \times 4K = 8K$ bits

How many entries are in a (2,2) predictor with the same number of bits?

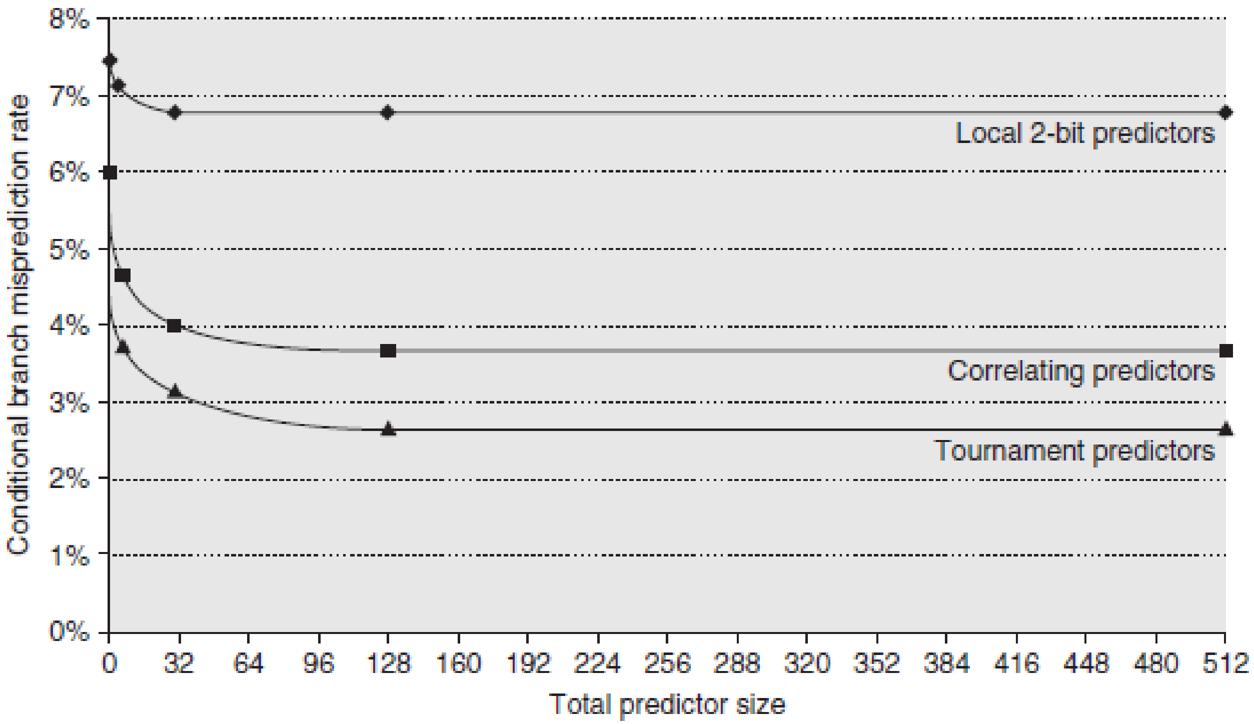
$2^2 \times 2 \times$ Number of prediction entries selected by branch = 8K

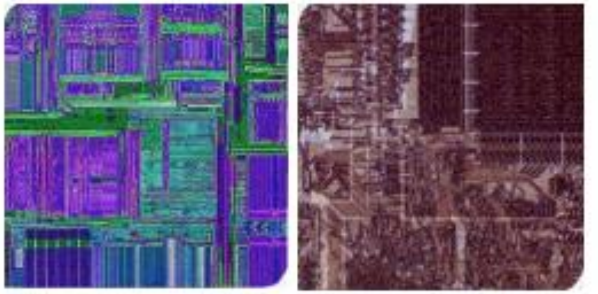
Hence, the number of prediction entries selected by branch = 1K.

Tournament Predictors

- Adaptively Combining Local and Global Predictors
- Use multiple predictors, usually one based on global information and one based on local information, and combining them with a selector.
- The advantage of a tournament predictor is its ability to select the right predictor for a particular branch, which is particularly crucial for the integer benchmarks.

Tournament Predictors





Dynamic Scheduling