Dynamic Branch Prediction

- In computer architecture, a **branch predictor** is a digital circuit that tries to guess which way a branch will go before this is known definitively. The **purpose of the branch predictor is to improve the flow in the instruction pipeline**. Branch predictors play a critical role in achieving **high effective performance** in many modern pipelined microprocessor architectures.
- Two classes:
  - static branch prediction (based on information avaiable at compile time, low cost)
  - dynamic branch prediction (based on program behavior)
- **Static branch prediction**
  - Static prediction is the **simplest** branch prediction technique because **it does not rely on information about the dynamic history of code executing**. Instead, it predicts the outcome of a branch based solely on the branch instruction.

Dynamic branch prediction

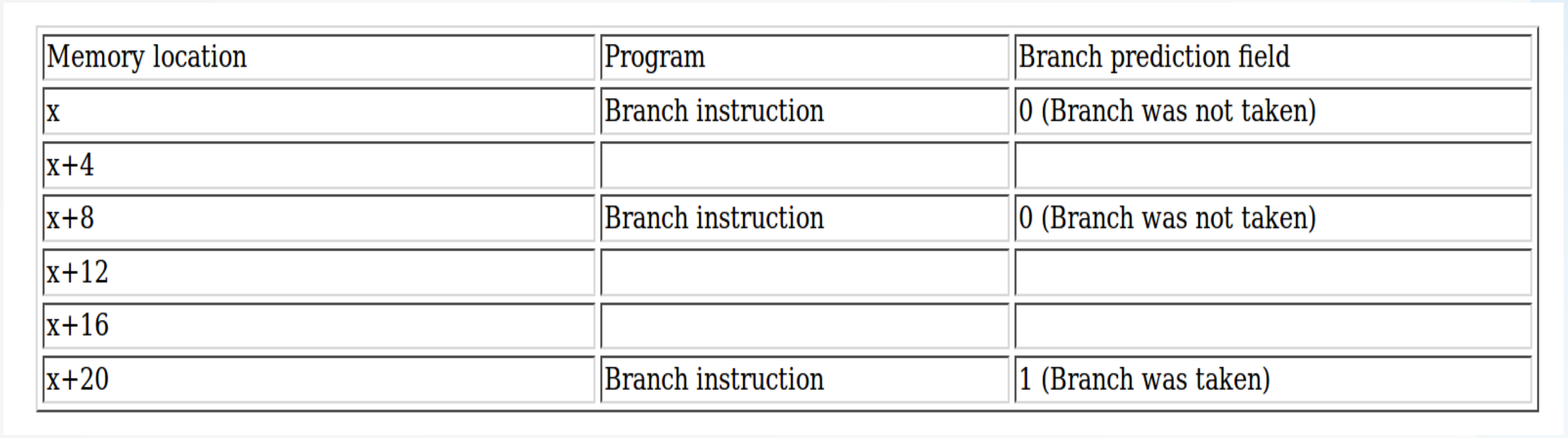
- Dynamic branch prediction uses information about taken or not taken branches gathered at run-time to predict the outcome of a branch.

1-bit predictor

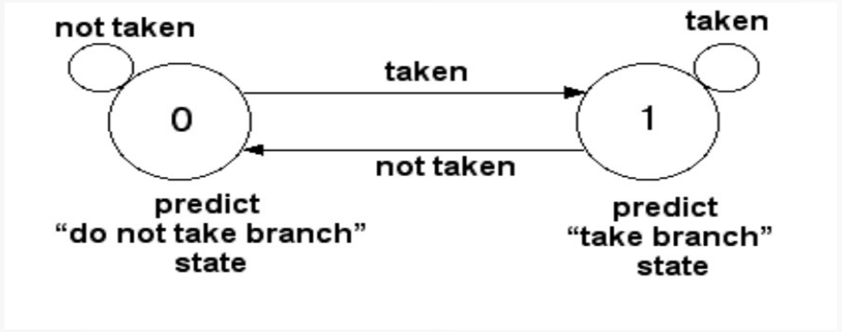
- It records the last outcome of the branch. This is the most simple version of dynamic branch predictor possible, although it is not very accurate.

- Branch instructions in the instruction cache include a branch prediction field that is used to predict if the branch should be taken.

- Use result from last time this instruction executed.



Memory location	Program	Branch prediction field
x	Branch instruction	0 (Branch was not taken)
x+4		
x+8	Branch instruction	0 (Branch was not taken)
x+12		
x+16		
x+20	Branch instruction	1 (Branch was taken)


not taken
taken
taken
0
1
not taken
predict "do not take branch" state
predict "take branch" state

In the simplest case, the field is a 1-bit tag:
0 <=> branch was not taken last time -or- predict "do not take branch" state
1 <=> branch was taken last time -or- predict "take branch" state

1-bit Predictor

- Set bit to 1 or 0:
  - Depending on whether branch Taken (T) or Not-taken (NT)
  - Pipeline checks bit value and predicts
  - If incorrect then need to discard speculatively executed instruction
- Actual outcome used to set the bit value.

Example

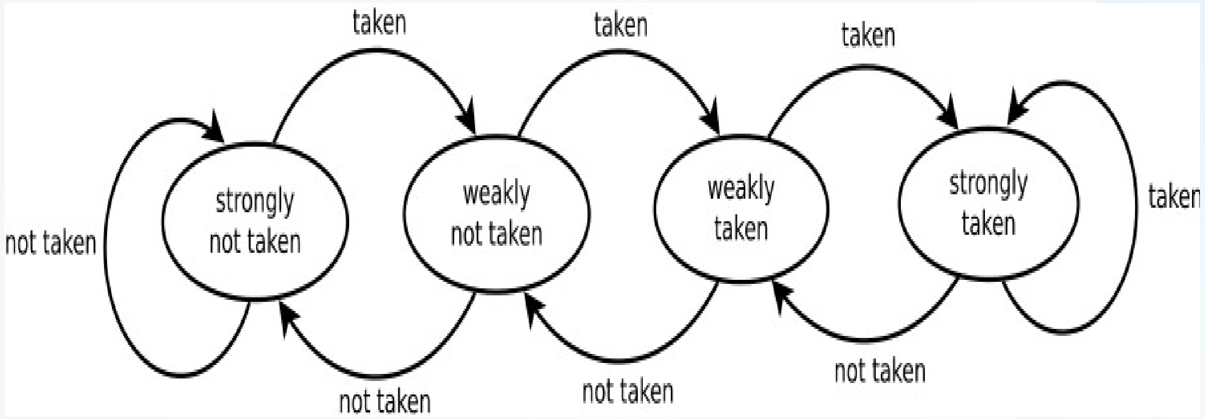
- Let initial value = T,
  - actual outcome of branches is-
  - Predictions are:

$$\begin{tikzcd}
\text{NT,} \arrow[dr] & \text{NT,} \arrow[dr] & \text{NT,} \arrow[dr] & \text{T,} \arrow[dr] & \text{T,} \arrow[dr] & \text{T} \\
\text{T,} & \text{NT,} & \text{NT,} & \text{NT,} & \text{T,} & \text{T}
\end{tikzcd}$$

    - 2 wrong (in red), 4 correct = 66% accuracy
- Limitation of 1 bit predictor:
  - Even if branch is almost always taken, we will be wrong at least twice
  - if branch alternates between taken, not taken
- 2-bit predictors can do even better
- In general, can have k-bit predictors.

A **2-bit predictor** is a state machine with four states:

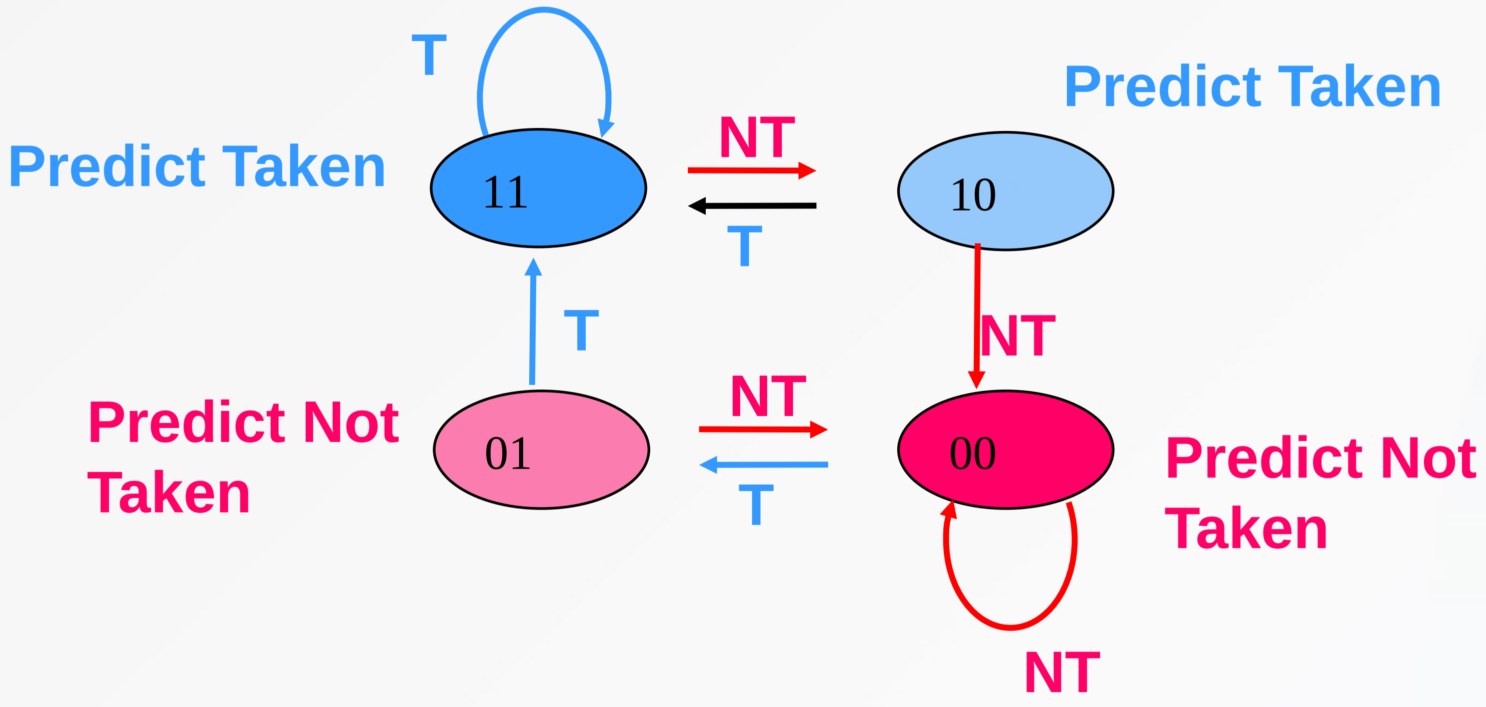
Weakly not taken(01)
Weakly taken(10)
Strongly not taken(00)
Strongly taken(11)



When a branch is evaluated, the corresponding state machine is updated. Branches evaluated as not taken change the state toward strongly not taken, and branches evaluated as taken change the state toward strongly taken. The advantage of the two-bit counter scheme over a one-bit scheme is that a conditional jump has to deviate twice from what it has done most in the past before the prediction changes. For example, a loop-closing conditional jump is mispredicted once rather than twice

2-bit Dynamic Branch Prediction Scheme

- Change prediction only if *twice* mispredicted:


T
Predict Taken
11
NT
T
10
Predict Taken
T
NT
Predict Not Taken
01
NT
T
00
Predict Not Taken
NT