# LECTURE - 14

# Dealing with Control Hazards

- Software techniques:
  - Branch delay slots
  - Software branch prediction
    - Canceling or nullifying branches
  - Misprediction rates can be high
  - Worse if multiple issue per cycle
- Hence, hardware/dynamic branch prediction

# Branch Prediction Buffer

- PC --> Taken/Not-Taken (T/NT) mapping
- Can use just the last few bits of PC
  - Prediction may be that of some other branch
  - Ok since correctness is not affected
- Shortcoming of this prediction scheme:
  - Branch mispredicted twice for each execution of a loop
  - Bad if loop is small

```
for(int i = 0; i < 10; i++) {
    x[i] = x[i] + C;
}
```

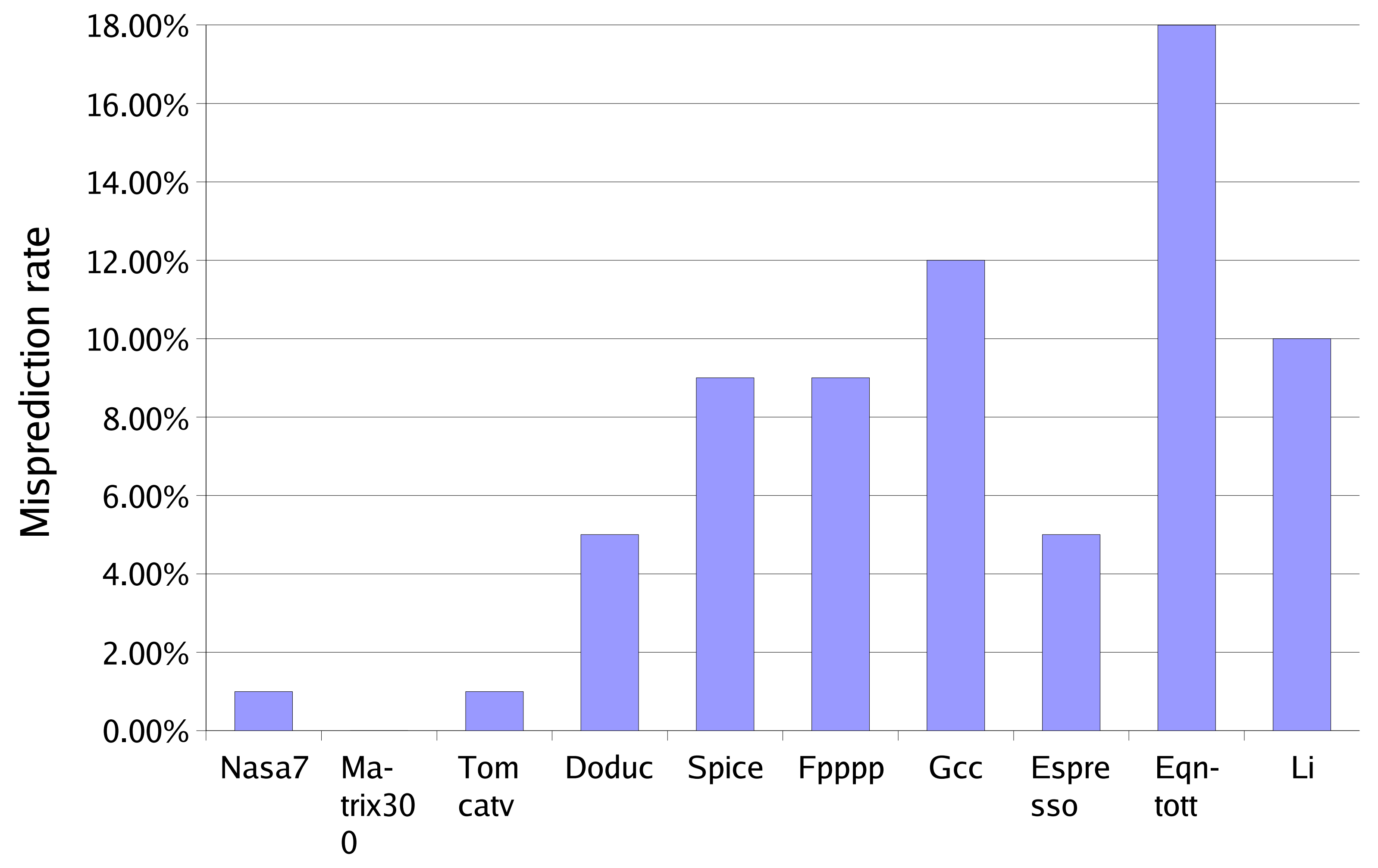
# Two-Bit Predictor

- Have to mispredict twice before changing prediction
    - Built in hysteresis
- General case is an n-bit predictor
    - 0 to (2^n)-1 saturating counter
    - 0 to (2^[n-1])-1 predict as taken
    - 2^[n-1] to (2^n)-1 predict as not-taken
- Experimental studies: 2-bit as good as n-bit

# Implementing Branch Prediction Buffers

- Implementing branch prediction buffers
  - Small cache accessed along with the instruction in IF
  - Or, additional 2 bits in instruction cache
- Note: branch prediction buffer not useful for DLX pipeline
  - Branch target not known earlier than branch condition

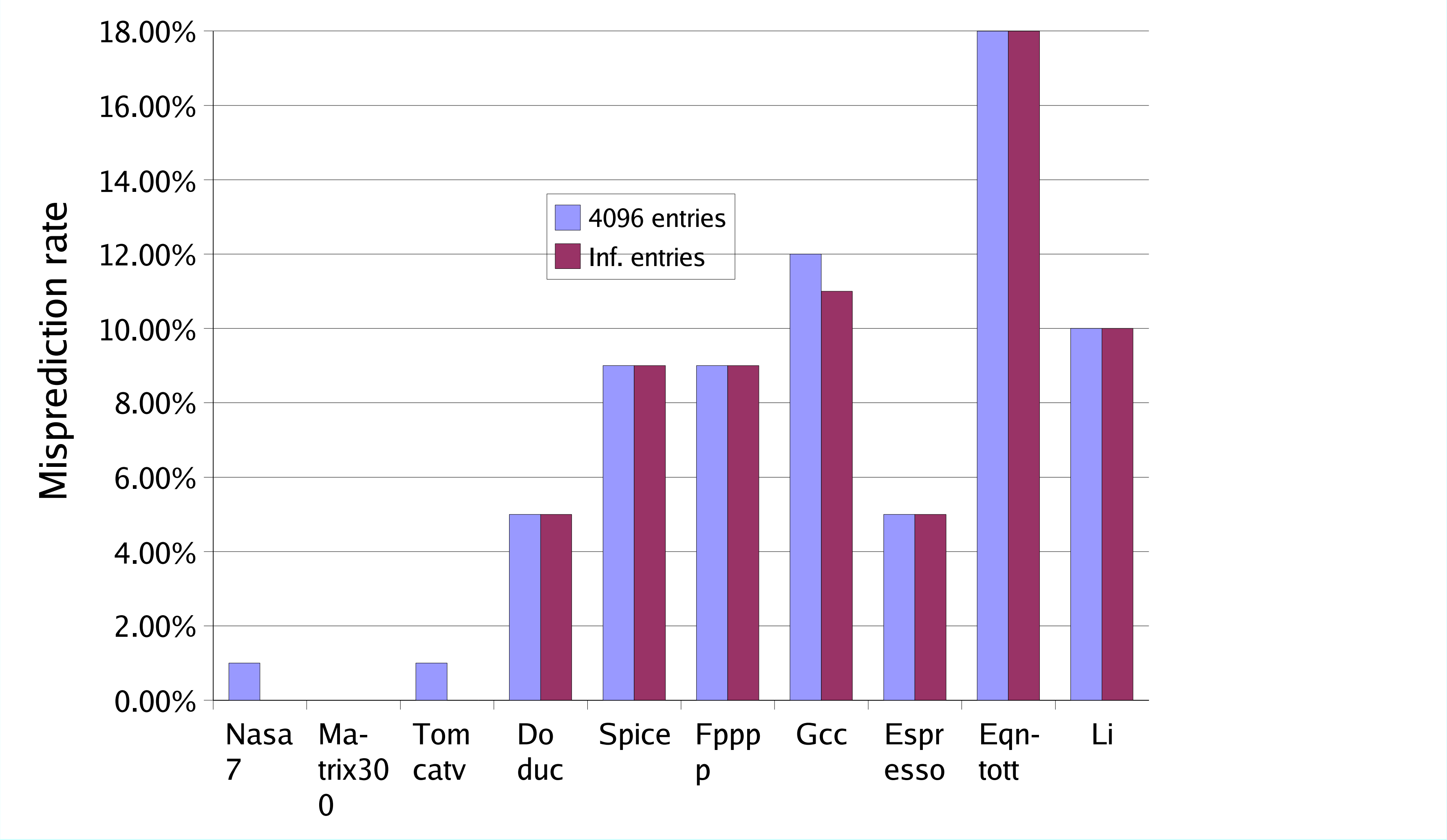
# Prediction Performance



- 4096 entries in the prediction buffer
- SPEC89, IBM Power architecture

# Improving Branch Prediction

- Two ways: increase buffer size, improve accuracy



# Improving Prediction Accuracy

- Predict branches based on outcomes of *recent other branches*

```
if(aa == 2) {
    aa = 0;
}
if(bb == 2) {
    bb = 0;
}
if(aa == bb) {
    // Do something
}
```

- Correlating, or two-level predictor

# Two-Level Predictor

- There are effectively two predictors for each branch:
  - Depending on whether previous branch is T/NT

| Prediction bits | Prediction if last branch NT | Prediction if last branch T |
|---|---|---|
| NT/NT | NT | NT |
| NT/T | NT | T |
| T/NT | T | NT |
| T/T | T | T |

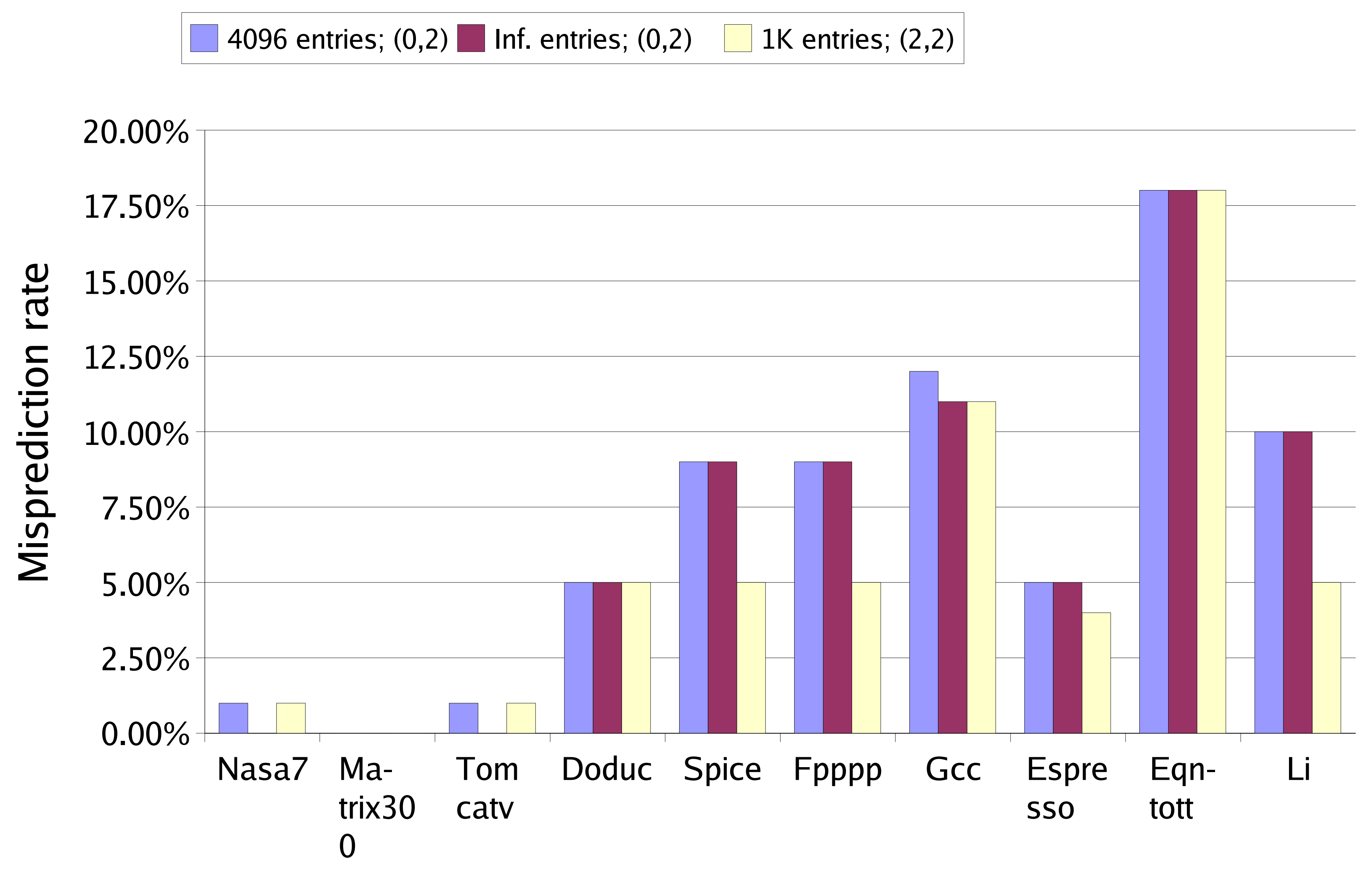
# Two-Level Predictor (continued)

- Last predictor was a (1,1) predictor
  - One bit each of history, and prediction
- General case is (m,n) predictor
  - m bits of history, n bits of prediction
- How to implement?
  - Have an m-bit shift register

# Cost of Two-Level Predictor

- Number of bits required:
  - Num. branch entries x 2^m x n
- How many bits in 4096 (0,2) predictor?
  - 8K
- How many branch entries for an 8K (2,2) predictor?
  - 1K

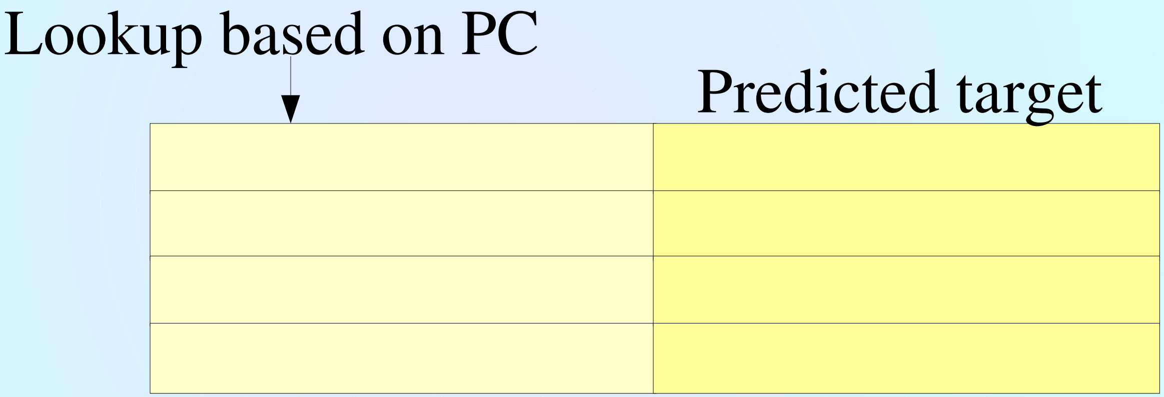
# Performance of (2,2) Predictor



# Branch Target Buffer

- Branch prediction buffer is not useful for DLX
  - Need to know target address by the end of IF
- Store branch target address also
  - Branch target buffer, or cache
- Access branch target buffer in IF cycle
  - Hit ==> predicted branch target known at the end of IF
  - We also need to know if the branch is predicted T/NT

# Branch Target Buffer (continued)

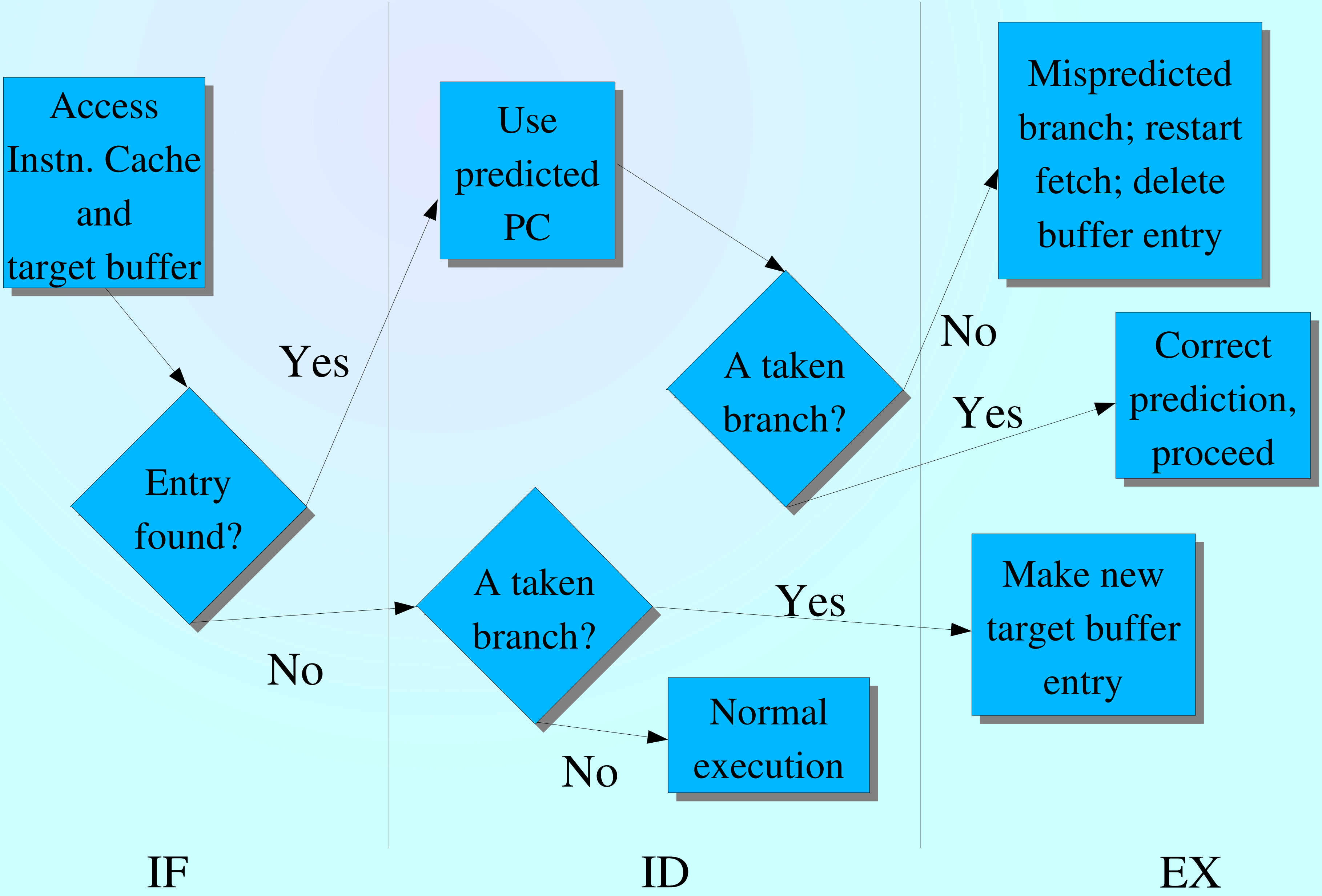


Lookup based on PC

Predicted target

- No entry found ==> (Target = PC+4)
- Exact match of PC is important
  - Since we are predicting even before knowing that it is a branch instruction
  - Hardware is similar to a cache
- Need to store predicted PC only for taken predictions

# Steps in Using a Target Buffer



Access Instn. Cache and target buffer

Entry found?

Yes

No

Use predicted PC

A taken branch?

No

Yes

Mispredicted branch; restart fetch; delete buffer entry

Correct prediction, proceed

A taken branch?

Yes

No

Make new target buffer entry

Normal execution

IF

ID

EX

# Penalties in Branch Prediction

| Buffer hit? | Branch taken? | Penalty |
|---|---|---|
| Yes | Yes | 0 |
| Yes | No | 2 |
| No | - | 2 |

- Given a prediction accuracy of *p*, a buffer hit-rate of *h*, and a taken branch frequency of *f*, what is the branch penalty?
  - $h \times (1-p) \times 2 + (1-h) \times f \times 2$

# Storing Target Instructions

- Directly store instructions instead of target address
  - Target buffer access is now allowed to take longer
  - Or, *branch folding* can be achieved
    - Replace fetched instruction with that found in the target buffer entry
    - Zero cycle unconditional branch; may be conditional as well