This Set

-1

Material from Section 4.3

This set under construction.

Outline

- Branch Prediction Overview
- Bimodal (One-Level) Predictor
- Correlating (Two-Level) Predictors: local, global, gshare
- Other topics to be added.
- Sample Problems

Practice Problems

Use problems below to practice material in this set.

Some solutions are detailed and are useful for understanding material.

Analysis Problems

2017 fep3a: TNTTnnn TnTTtttt bimodal, var pattern len. local. Hist sz. GHR.

2016 fep3a: B2: TNTNTNT N (nn or tt) bimodal. local. min LH

2013 fep3: B1: TNTTTN, B2: TNTrTN, B3: T.. bimodal, local. PHT colli. GHR

2014 fep3: TTTNN, B2: rrrqqq (grps of 3) bimodal. local GHR val

2015 fep3: NTTNNN, B2: T2,4,6NNNN, B3: T.. bimodal, local, min GHR siz

Branch Predictor Variations, and Hardware

2016 fep3 (b) Post-loop branch on global predictor variations.

2017 fep3 (b): Convert illustrated bimodal into local predictor.

2013 fep3b: Draw a digram of local predictor.

Branch and Target Prediction

-4

Motivation

Branches occur frequently in code.

At best, one cycle of branch delay; more with dependencies.

Therefore, impact on CPI is large.

Techniques

Branch Direction Prediction:
Predict outcome of branch. (Taken or not taken.)

Branch Target Prediction:
Predict branch or other CTI's target address.

Note: *CTI or Control Transfer Instruction*, is any instruction that causes execution to go somewhere else, such as a branch, jump, or trap.

Branch Direction Prediction

-5

Methods Covered

Simple, One-Level, Predictor

bimodal, a.k.a. *One-level predictor*

Commonly used in simpler CPUs.

Correlating (Two-Level) Predictors

Local History, a.k.a. *PAg*.

Global History, a.k.a. *GAg*.

gshare.

Commonly used in general purpose CPUs.

Branch Prediction Idea

Idea: Predict using past behavior.

Example:

```
LOOP:
 lw    r1, 0(r2)      # Load random number, either 0 or 1.
 addi  r2, r2, 4
 slt   r6, r2, r7
 beq   r1, r0 SKIP    # T N N T   N T T T N     # Random, no pattern.
 nop
 addi  r3, r3, 1
SKIP:
 bne   r6, r0 LOOP    # T T T ... T N T T T     # 99 T's, 1 N, 99 T's, ...
 nop
```

Second branch, `bne`, taken 99 out of 100 executions.

Pattern for `bne`: T T T . . . T N T T T

First branch shows no pattern.

Prediction Accuracy

SPEC89 benchmarks on IBM POWER (predecessor to PowerPC).

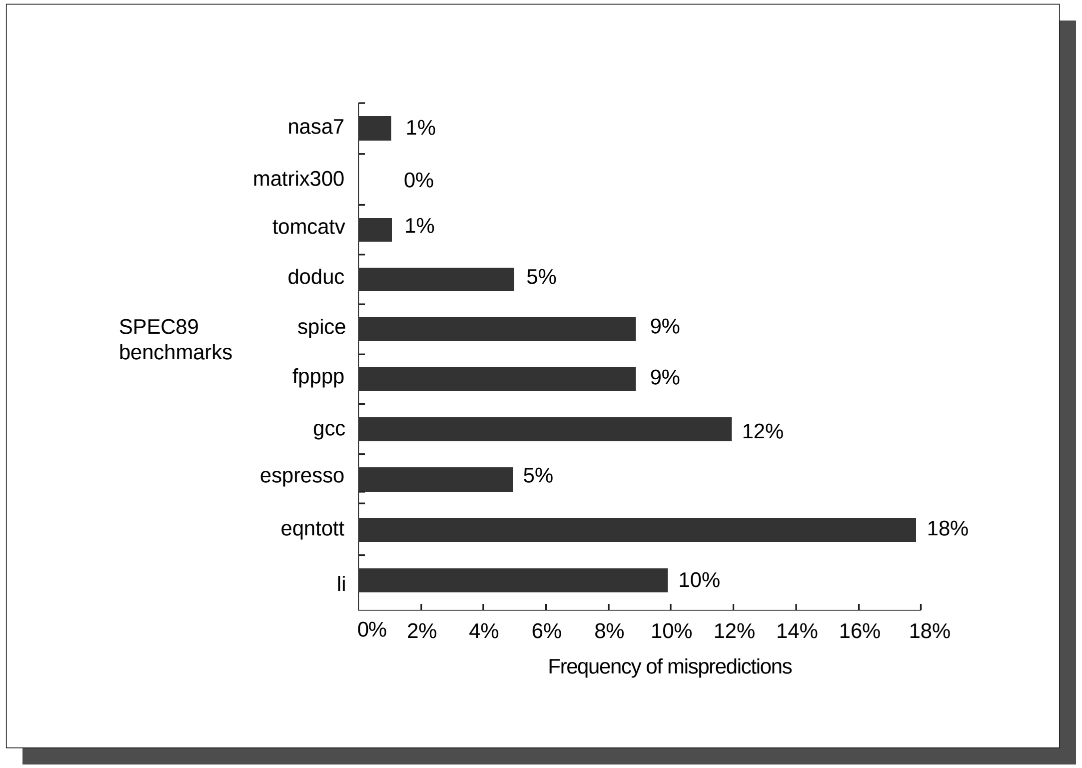


FIGURE 4.14 Prediction accuracy of a 4096-entry two-bit prediction buffer for the SPEC89 benchmarks.

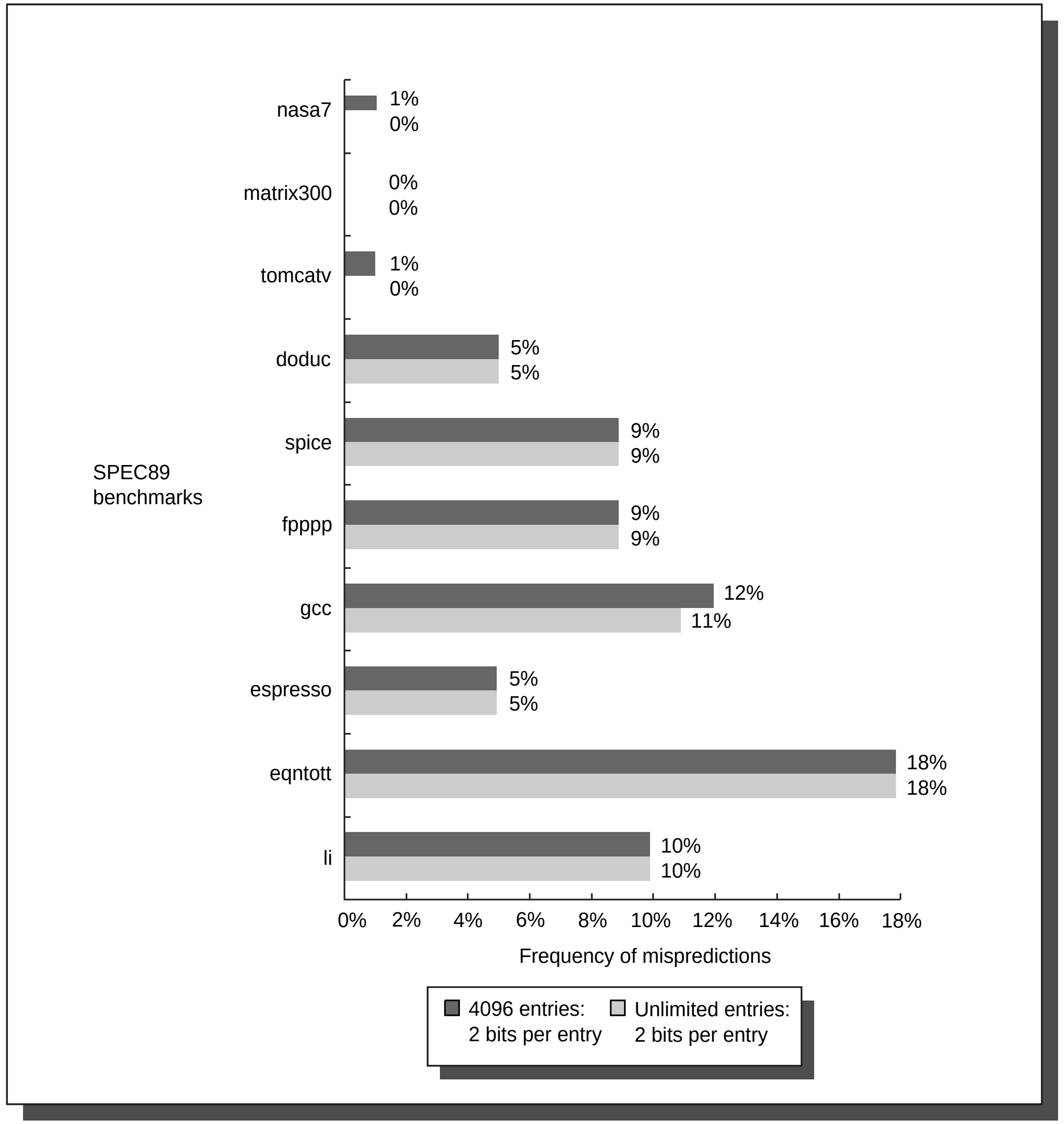


FIGURE 4.15 Prediction accuracy of a 4096-entry two-bit prediction buffer versus an infinite buffer for the SPEC89 benchmarks.

Branch Prediction Terminology

-8

Outcome: [of a branch instruction execution].
Whether the branch is taken or not taken.

T:
A taken branch. Used in diagrams to show branch outcomes.

N:
A branch that is not taken. Used in diagrams to show branch outcomes.

Prediction:
The outcome of a branch predicted by a branch predictor.

Branch Prediction Terminology

-9

Resolve: [a branch].
To determine whether a branch is taken and if so, to which address. In our 5-stage MIPS this is done in ID.

Misprediction:
An incorrectly predicted branch.

Prediction Accuracy: [of a branch prediction scheme].
The number of correct predictions divided by the number of predictions.

-9

Branch Prediction Terminology (Continued)

-10

Speculative Execution:

The execution of instructions which may not be on the correct program path (due to a predicted CTI) or which may not be correct for other reasons (such as load/store dependence prediction [a topic that is usually not covered in this class]).

Misprediction Recovery:

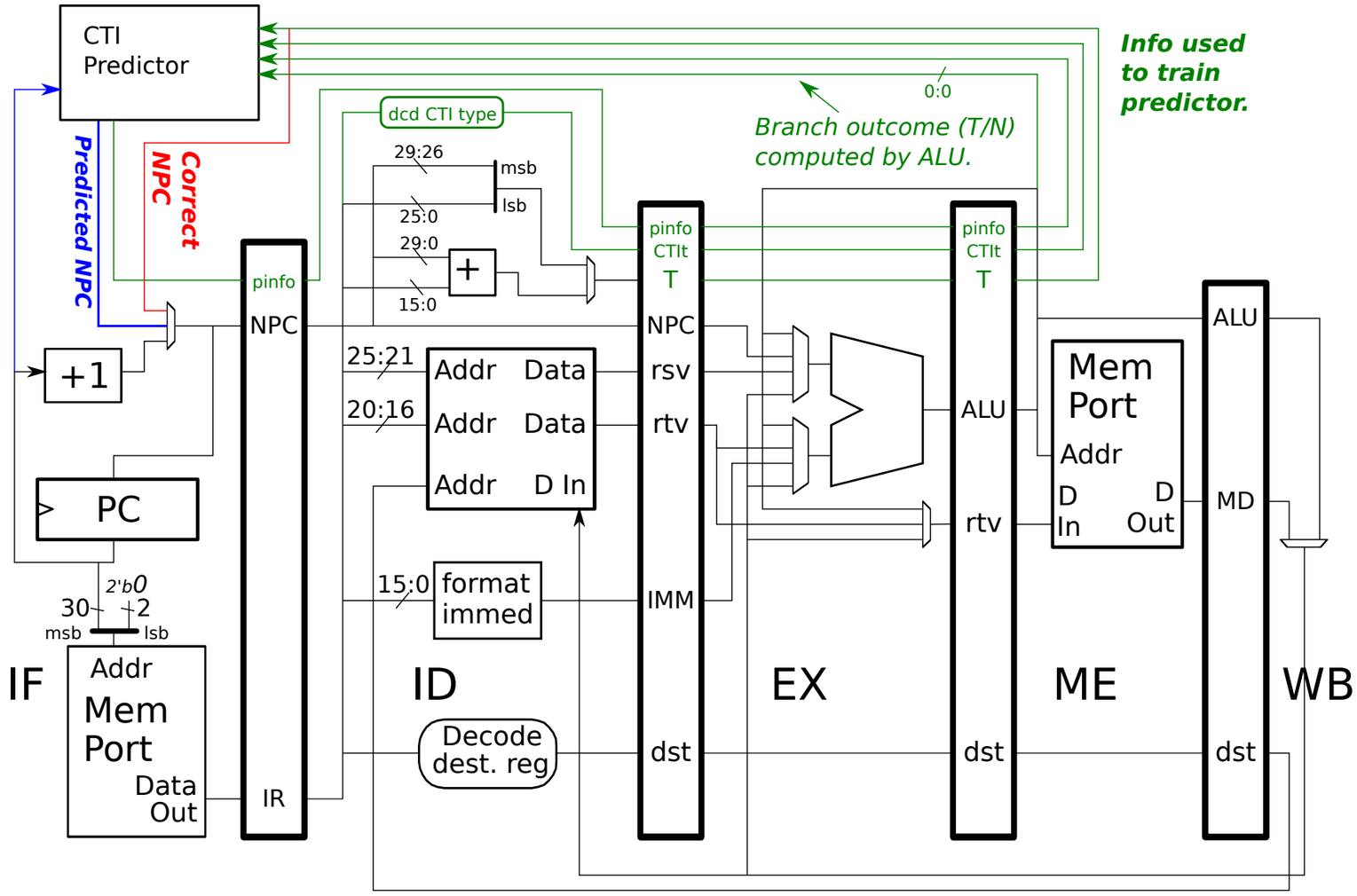
Undoing the effect of speculatively executed instructions . . .

. . . and re-starting instruction fetch at the correct address.

-10

-11

Hardware Overview



-11

Bimodal Branch Predictor

-12

Bimodal Branch Predictor:

A branch direction predictor that associates a 2-bit counter (just a 2-bit unsigned integer) with each branch. The counter is incremented when the branch is taken and decremented when the branch is not taken. The branch is predicted taken if the counter value is 2 or 3.

Example of 2-Bit Counter Used for Four-Iteration Loop

In diagram below initial counter value assumed to be zero.

```
# Counter:        0  1  2  3  2   3  3  3  2   3  3  3  2   3  3  3  2
beq r1, r2, TARG   T  T  T  N   T  T  T  N   T  T  T  N   T  T  T  N
# Prediction      n  n  t  t   t  t  t  t   t  t  t  t   t  t  t  t
# Outcome:         x  x     x           x           x           x
```

Prediction Accuracy: $\frac{3}{4}$, based on repeating pattern.

-12

-13

Bimodal Branch Predictor

Characteristics:

Low cost.

Used in many 20th century processors.

-13

Bimodal Branch Predictor

-14

Bimodal Branch Predictor

Idea: maintain a *branch history* for each branch instruction.

Branch History:
Information about past behavior of the branch.

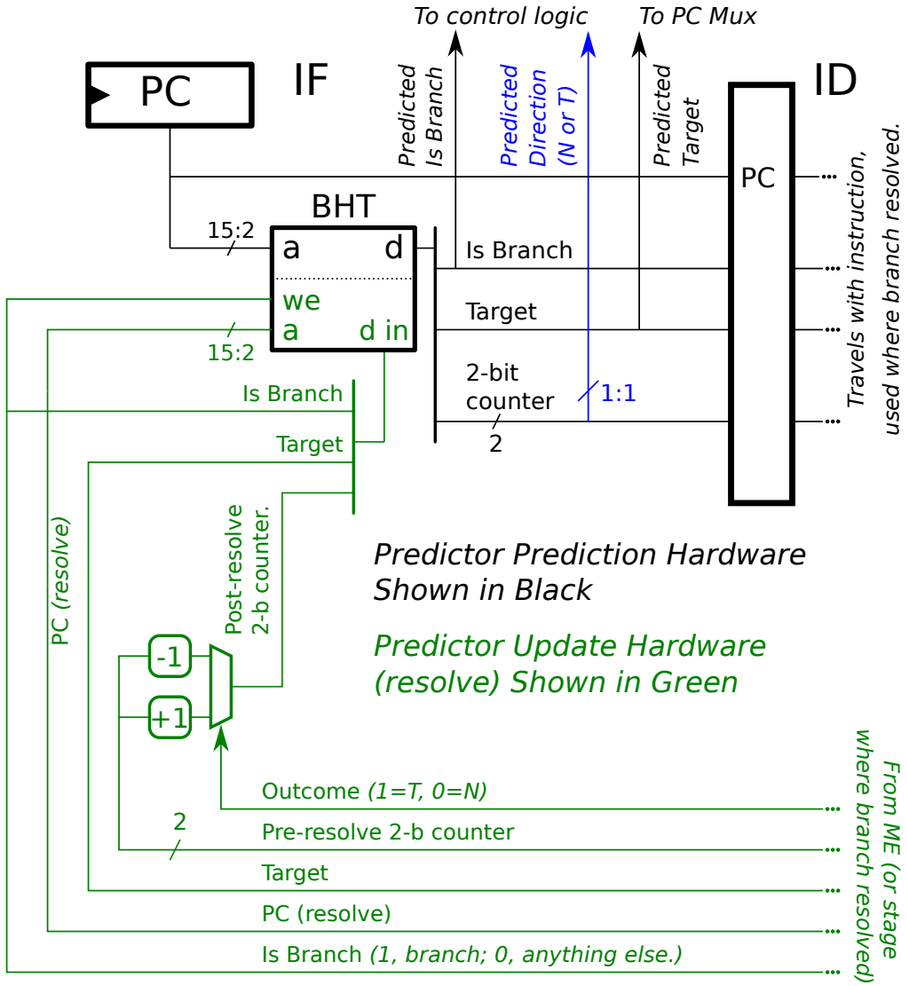
Branch histories stored in a *branch history table (BHT)*.

Often, branch history is sort of number of times branch taken. . .
. . . minus number of times not taken.

Other types of history possible.

Branch history read to make a prediction.

Branch history updated when branch outcome known.



-14

Branch History Counter

-15

Branch History Counter and *Two-Bit Counter*

If a counter used, branch history incremented when branch taken...
... and decremented when branch not taken.

Symbol n denotes number of bits for branch history.

To save space and for performance reasons ...
... branch history limited to a few bits, usually $n = 2$.

Branch history updated using a *saturating counter*.

A saturating counter is an arithmetic unit that can add or subtract one ...
... in which $x + 1 \rightarrow x + 1$ for $x \in [0, 2^n - 2]$ ...
... $x - 1 \rightarrow x - 1$ for $x \in [1, 2^n - 1]$ ...
... $(2^n - 1) + 1 \rightarrow 2^n - 1$ ...
... and $0 - 1 \rightarrow 0$.

For an n-bit counter, predict taken if counter $\geq 2^{n-1}$.

-15

Branch History (2-Bit) Counter Example

-16

Example of 2-Bit Counter Used for Four-Iteration Loop

In diagram below initial counter value assumed to be zero.

```
# Counter:        0  1  2  3  2   3  3  3  2   3  3  3  2   3  3  3  2
beq r1, r2, TARG   T  T  T  N   T  T  T  N   T  T  T  N   T  T  T  N
# Prediction      n  n  t  t   t  t  t  t   t  t  t  t   t  t  t  t
# Outcome:        x  x     x         x           x           x
```

Prediction Accuracy: $\frac{3}{4}$, based on repeating pattern.

-16

One-Level Branch Predictor Hardware

-17

Bimodal aka *One-Level Branch Predictor* Hardware

Illustrated for 5-stage MIPS implementation . . .
. . . even though prediction not very useful.

Branch Prediction Steps

1: *Predict*.

Read branch history, available in IF.

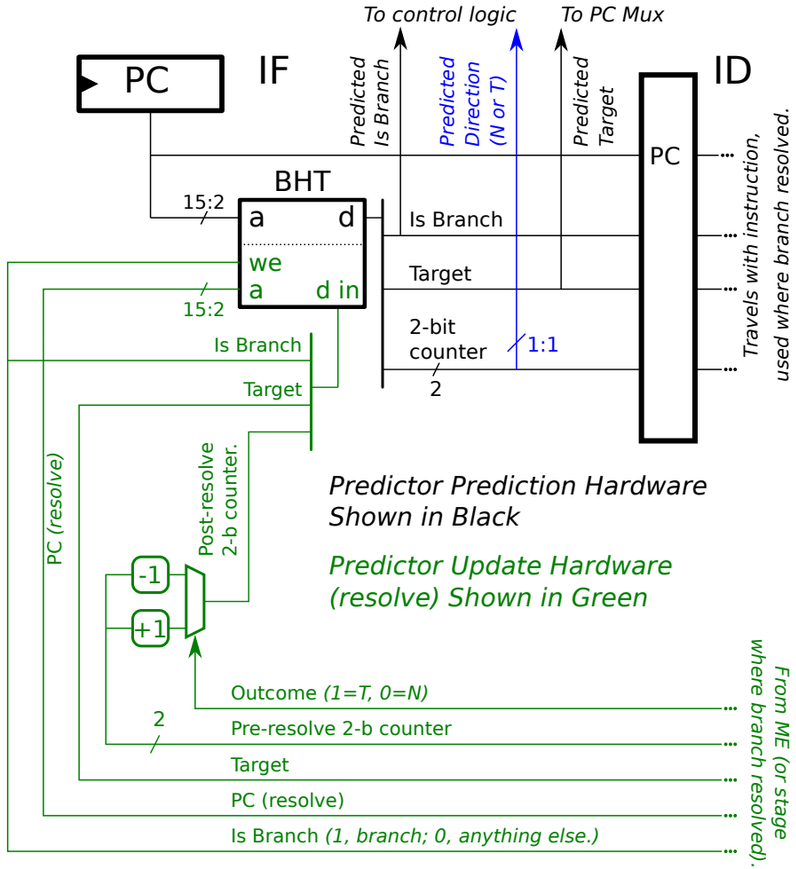
2: *Resolve* (Determine Branch Outcome)

Execute predicted branch in usual way.

3: *Recover* (If necessary.)

Undo effect of speculatively executing instructions, start fetching from correct path.

4: *Update* Branch History



-17

Branch History Table

-18

Branch History Table

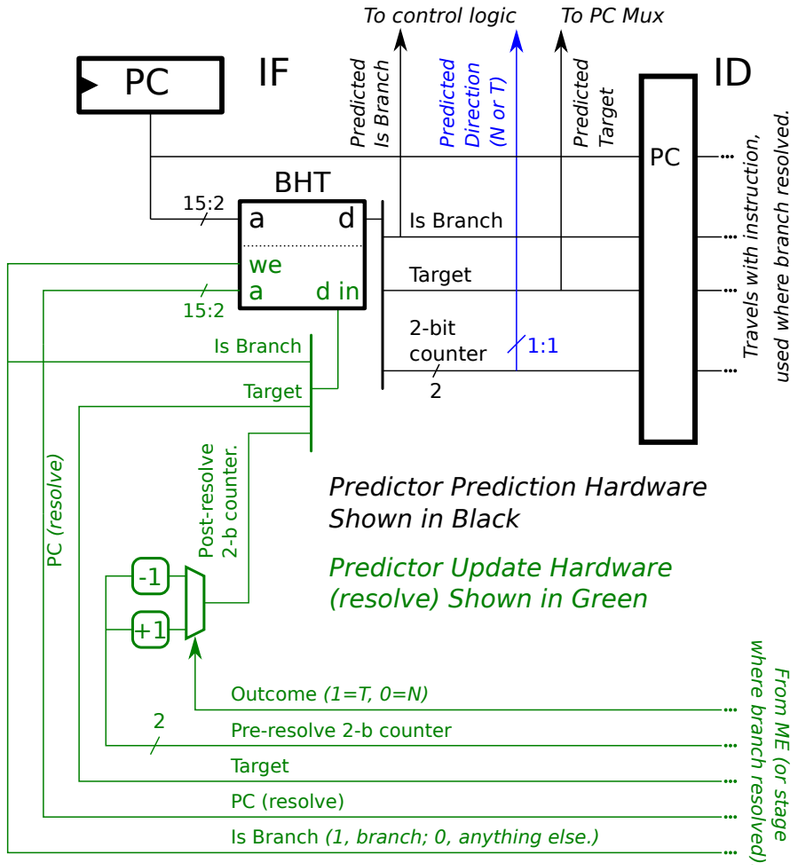
Stores info about each branch.

Used in all branch predictors, the info varies based on predictor type.

Implemented using a memory device.

Address (called index) is hash of branch address (PC).

For 2^m-entry BHT, hash is m lowest bits of branch PC **skipping alignment**.



LSU EE 4720 Lecture Transparency. Formatted 13:42, 12 April 2019 from lsli12-TeXize.

-18

-19

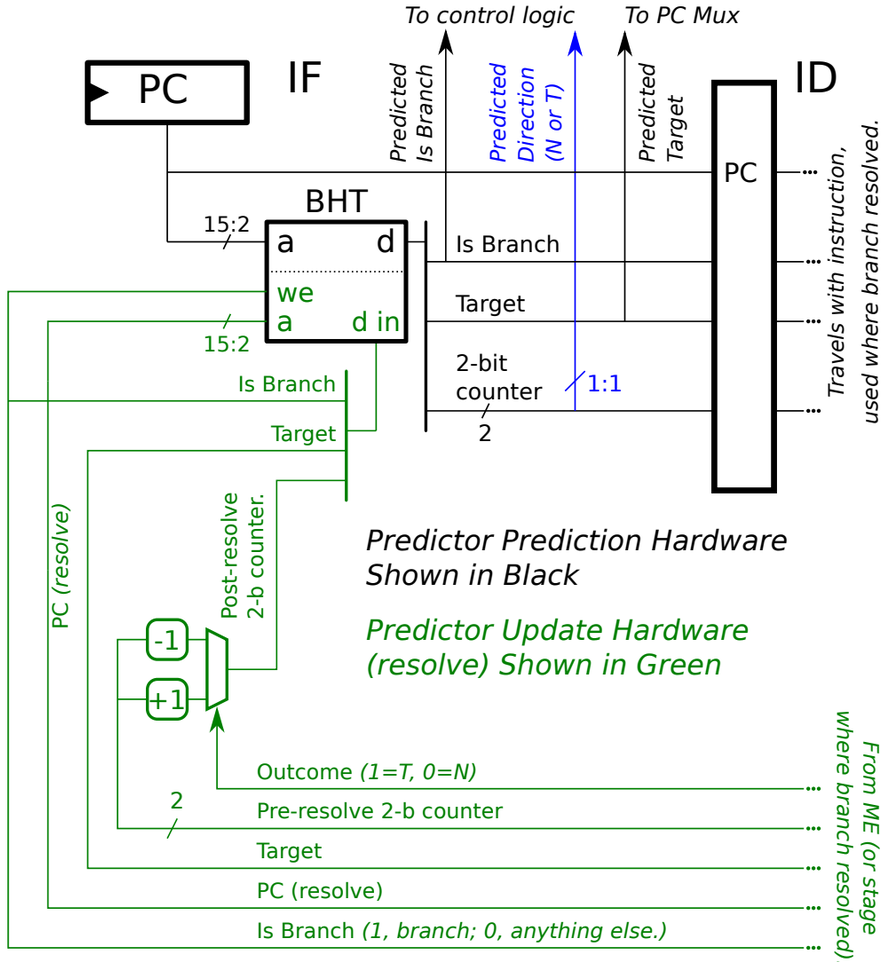
Output of BHT

CTI Type, indicating whether insn is a branch, jump, etc.

Note: *CTI*, Control Transfer Instruction, is any instruction that causes execution to go somewhere else, such as a branch, jump, or trap.

Target Address, the address to go to if CTI taken.

Two-Bit Counter, bias in taken direction.



-19

Sample Local History

-20

Outcomes for individual branches, categorized by pattern, sorted by frequency.

Branches running TEX text formatter compiled for SPARC (Solaris).

```
Arbitrary, pat 60288, br732164, 0.7743 0.7170 0.7199  (0.19675)
          % Patterns # Branches gshre local  corr   Local History
  0:     fe7f  0.0004     1397  0.912 0.916 0.896  TTTTTTTNNTTTTTTT  0
  1:     ff3f  0.0004     1323  0.924 0.909 0.900  TTTTTTNNTTTTTTTT  0
  2:     fcff  0.0004     1317  0.949 0.939 0.948  TTTTTTTTNNTTTTTT  0
  3:     ff9f  0.0003     1245  0.910 0.905 0.898  TTTTTNNTTTTTTTTT  0
  4:     f9ff  0.0003     1235  0.955 0.950 0.955  TTTTTTTTTNNTTTTT  0
  5:     ffcf  0.0003     1188  0.926 0.921 0.923  TTTTNNTTTTTTTTTT  0
  6:       60  0.0003     1163  0.873 0.829 0.854  NNNNNTTNNNNNNNNN  0
  7:      180  0.0003     1159  0.955 0.914 0.926  NNNNNNNTTNNNNNNN  0
  8:      300  0.0003     1158  0.949 0.926 0.934  NNNNNNNNTTNNNNNN  0
  9:       c0  0.0003     1155  0.944 0.917 0.926  NNNNNNTTNNNNNNNN  0
```


-20

-21

```
Short Loop, pat 124, br  137681, 0.8908 0.9055 0.7441  (0.03700)
          % Patterns # Branches gshre local  corr   Local History
  0:     5555  0.0040    14753  0.987 0.981 0.912  TNTNTNTNTNTNTNTN  1
  1:     aaaa  0.0040    14730  0.859 0.978 0.461  NTNTNTNTNTNTNTNT  1
  2:     9249  0.0022     8062  0.997 0.992 0.988  TNNTNNTNNTNNTNNT  1
  3:     4924  0.0022     8055  0.997 0.998 0.998  NNTNNTNNTNNTNNTN  1
  4:     2492  0.0022     8047  0.993 0.991 0.009  NTNNTNNTNNTNNTNN  1
  5:     db6d  0.0013     4864  0.713 0.915 0.065  TNTTNTTNTTNTTNTT  1
  6:     b6db  0.0013     4713  0.862 0.903 0.926  TTNTTNTTNTTNTTNT  1
  7:     6db6  0.0012     4640  0.991 0.978 0.970  NTTNTTNTTNTTNTTN  1
  8:     bbbb  0.0008     3061  0.896 0.936 0.949  TTNTTTNTTTNTTTNT  1

Long Loop?, pat 32, br  185795, 0.9170 0.9052 0.9096  (0.04993)
  0:     fffe  0.0025     9204  0.902 0.930 0.913  NTTTTTTTTTTTTTTT  2
  1:     8000  0.0025     9198  0.654 0.700 0.705  NNNNNNNNNNNNNNNT  2
  2:     7fff  0.0022     8052  0.890 0.817 0.818  TTTTTTTTTTTTTTTN  2
  3:     ffbf  0.0018     6800  0.933 0.908 0.920  TTTTTTNTTTTTTTTT  2
  4:     feff  0.0018     6782  0.946 0.938 0.942  TTTTTTTTNTTTTTTT  2
  5:     ff7f  0.0018     6778  0.949 0.946 0.950  TTTTTTTNTTTTTTTT  2
  6:     fdff  0.0018     6738  0.947 0.941 0.946  TTTTTTTTTNTTTTTT  2
  7:        1  0.0018     6690  0.955 0.945 0.942  TNNNNNNNNNNNNNNN  2
  8:     fffd  0.0018     6667  0.968 0.966 0.967  TNTTTTTTTTTTTTTT  2
```


-21

-22

```
Phase Change, pat 26, br 48190, 0.8453 0.9040 0.8470  (0.01295)
          % Patterns # Branches gshre local  corr   Local History
  0:     c000  0.0012     4554  0.653 0.777 0.680  NNNNNNNNNNNNNNTT  3
  1:     e000  0.0009     3420  0.714 0.859 0.758  NNNNNNNNNNNNNTTT  3
  2:     f000  0.0008     2942  0.756 0.888 0.788  NNNNNNNNNNNNTTTT  3
  3:     fffc  0.0008     2878  0.908 0.960 0.959  NNTTTTTTTTTTTTTT  3
  4:     f800  0.0007     2642  0.786 0.917 0.827  NNNNNNNNNNNTTTTT  3
  5:        3  0.0007     2572  0.968 0.952 0.951  TTNNNNNNNNNNNNNN  3
  6:     fc00  0.0007     2435  0.815 0.933 0.854  NNNNNNNNNNTTTTTT  3
  7:     fe00  0.0006     2225  0.836 0.936 0.876  NNNNNNNNNTTTTTTT  3
  8:     ff00  0.0006     2140  0.856 0.947 0.931  NNNNNNNNTTTTTTTT  3
  9:     ff80  0.0006     2061  0.854 0.941 0.934  NNNNNNNTTTTTTTTT  3


One Way, pat    2, br  2617433, 0.9917 0.9934 0.9897  (0.70337)
  0:     ffff  0.5151  1916950  0.993 0.996 0.993  TTTTTTTTTTTTTTTT  4
  1:        0  0.1882   700483  0.988 0.986 0.982  NNNNNNNNNNNNNNNN  4
```


-22

Two-Level Correlating Predictors

-23

Idea: Base branch decision on . . .

. . . the address of the branch instruction (as in the one-level scheme) . . .

. . . and the most recent branch outcomes.

History:

The outcome (taken or not taken) of the most recent branches. Usually stored as a bit vector with 1 indicating taken.

Pattern History Table (PHT):

Memory for 2-bit counters, indexed (addressed) by some combination of history and the branch instruction address.

-24

Some Types of Two-Level Predictors

Global, a.k.a. *GAg*.

History is global (same for all branches), stored in a *global history register* (*GHR*).

PHT indexed using history only.

gshare

History is global (same for all branches), stored in a *global history register* (*GHR*).

PHT indexed using history exclusive-ored with branch address.

gselect

History is global (same for all branches), stored in a *global history register* (*GHR*).

PHT indexed using history concatenated with branch address.

-25

Local, a.k.a., *PAg*.

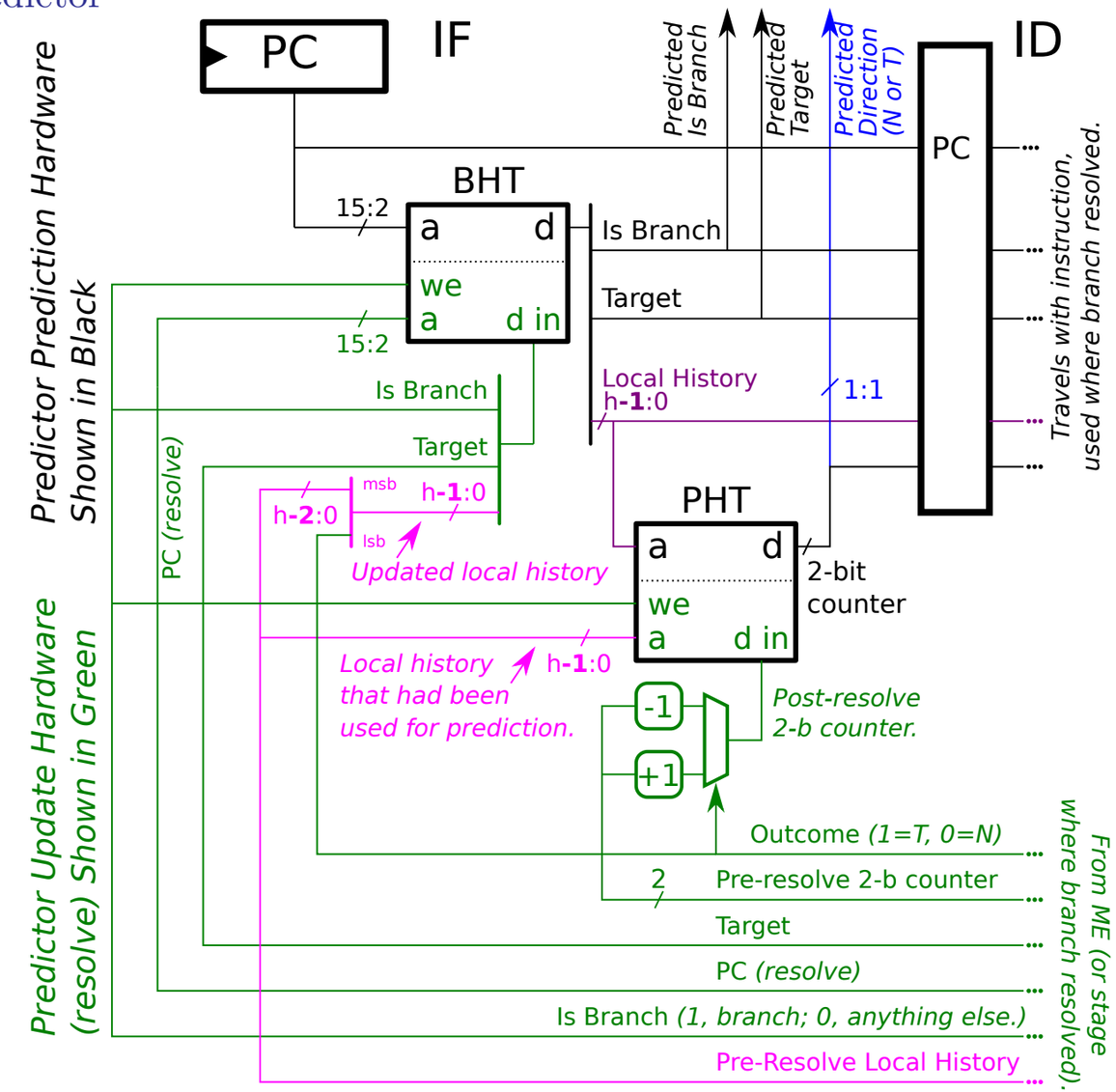
History is local, BHT stores history for each branch.

PHT indexed using history only.

-25

-26

h-Outcome Local History Predictor



-26

Global History Example

-27

```
# Loop always iterates 4 times.
# Branch below never taken.
 bne  r2, SKIP      N                       N
 add.d f0, f0, f2
SKIP:
 addi  r1, r0, 4
LOOP:
 mul.d f0, f0, f2
 bne   r1, LOOP          T   T   T   N    ... T   T   T   N  ...
 addi  r1, r1, -1
# Cycle                10  20  30  40  50  110 120 130 140 150
#
# Global History (m=4), X: depends on earlier branches.
# 10  XXXN  Human would predict taken.
# 20  XXNT  Human would predict taken.
# 30  XNTT  Human would predict taken.
# 40  NTTT  Human would predict not taken.
# 50  TTTN
```