Lecture 17

Computer Architecture
(Spring 2020)

Dynamic Hardware Branch Prediction

& Branch-Target Buffers

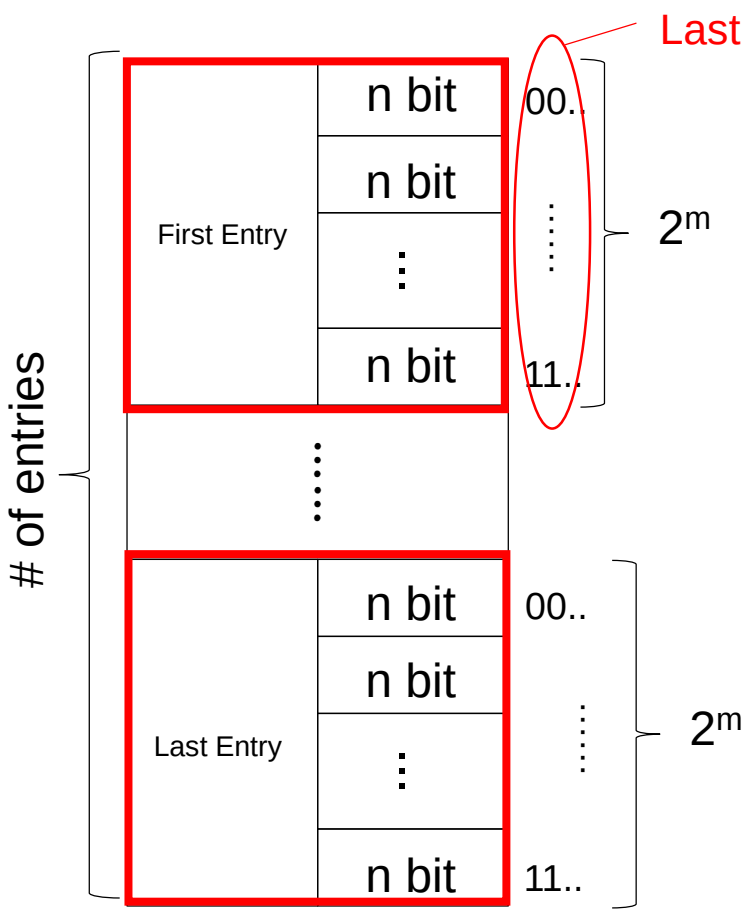
Dr. Duo Liu (刘铎)
Office: Main Building 0626
Email: liuduo@cqu.edu.cn

Correlating Branch Predictors

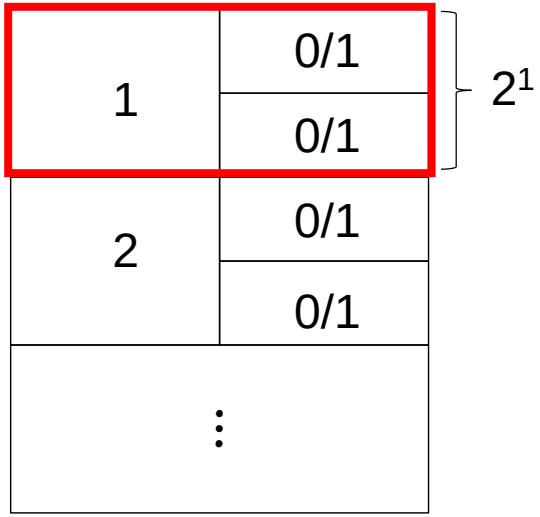
```
if (aa==2)
        aa=0;
if (bb==2)
        bb=0;
if (aa!=bb) {
```

- The limitation of the basic 2-bit predictor:
  - The 2-bit predictor schemes use only the recent behavior of **a single branch** to predict the future behavior of that branch. Increasing to 3-bit or more does not help much!
  - How about looking at the recent behavior of other branches?
  - Look the code on the top-right corner: if the first two branches are taken, the 3rd is never taken.
- Correlating predictor / 2-level predictor:
  - Adds information of the most recent branches to decide how to predict a given branch.
  - An (m,n) 2-level predictor uses the behavior of the last m branches to choose from 2^m branch predictors, each of which is an n-bit predictor for a single branch.
  - The size of BHT = # of Entries $\times$ # of bits / Entry = # of Entries $\times$ 2^m $\times$ n

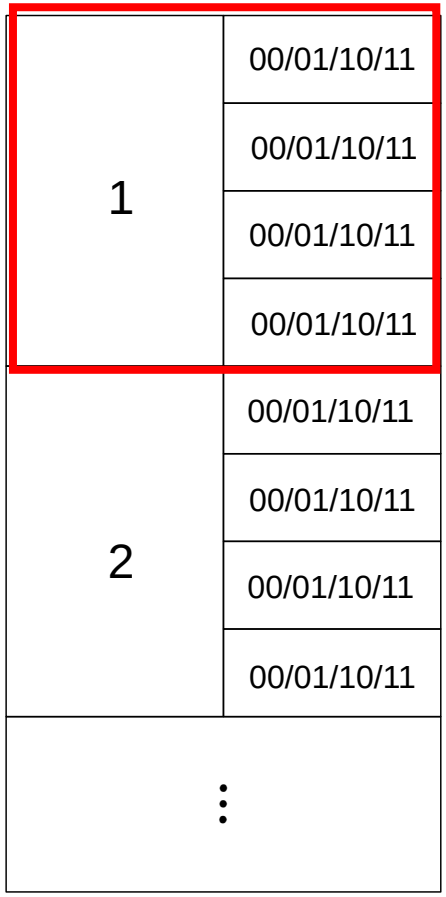
Correlating Branch Predictors



(m,n) correlating predictor BHT



(1,1) correlating predictor BHT



(2,2) correlating predictor BHT

Adding "Global" Information to Branch Prediction: Correlating (or Two-Level) Branch Predictors

```
if (a == 2)
  a = 0;
…
if (b == 2)
  b = 0;
…
if (a != b) {
  …
```

```
      DSUBUI R3, R1, #2
b1 →  BNEZ R3, L1
      DADD R1, R0, R0
L1:   DSUBUI R3, R2, #2
b2 →  BNEZ R3, L2
      DADD R2, R0, R0
L2:   DSUBU R3, R1, R2
b3 →  BEQZ R3, L3
```

- Prediction accuracy can be improved by accounting for branch spatial correlations and looking at the behavior of other branches
  - e.g. in the example above, if the first two branches (b1 and b2) are not taken then the third (b3) will be taken

A Simple Example: Set-Up

```
if (d == 0)
  d = 1;
if (d == 1) {
  …
```

```
b1 → BNEZ R1, L1
     DADDIU R1, R0, #1
L1:  DSUBUI R3, R1, #1
b2 → BNEZ R3, L2
     …
L2:  …
```

init d	d==0?	b1	d before b2	d==1?	b2
0	yes	not taken	1	yes	not taken
1	no	taken	1	yes	not taken
x ≠{0,1}	no	taken	x ≠ {0,1}	no	taken

Correlation: if b1 is not taken then d is set to 1 and b2 is also not taken

A Simple Example: 1-Bit Predictor

```
if (d == 0)
  d = 1;
if (d == 1) {
  …
```

Assumption: d alternates between 2 and 0

```
b1 ──→ BNEZ R1, L1
       DADDIU R1, R0, #1
L1:    DSUBUI R3, R1, #1
b2 ──→ BNEZ R3, L2
       …
L2:    …
```

d=?	b1 pred.	b1 action	b1 new pred.	b2 pred.	b2 action	b2 new pred.
2	NT	T	T	NT	T	T
0	T	NT	NT	T	NT	NT
2	NT	T	T	NT	T	T
0	T	NT	NT	T	NT	NT

A 1-bit predictor that is initialized to not-taken mispredicts all branches

A Simple Example: 1-Bit Predictor with 1-Bit Correlation {i.e., a (1,1) Predictor}

```
if (d == 0)
  d = 1;
if (d == 1) {
  …
```

```
b1 → BNEZ R1, L1
     DADDIU R1, R0, #1
L1:  DSUBUI R3, R1, #1
b2 → BNEZ R3, L2
     …
L2:  …
```

Assumption: d alternates between 2 and 0

X/Y: use X if last branch was not taken, use Y if last branch was taken

d=?	b1 pred.	b1 action	b1 new pred.	b2 pred.	b2 action	b2 new pred.
2	**NT**/NT	T	T/NT	NT/**NT**	T	NT/T
0	T/**NT**	NT	T/NT	**NT**/T	NT	NT/T
2	**T**/NT	T	T/NT	NT/**T**	T	NT/T
0	T/**NT**	NT	T/NT	**NT**/T	NT	NT/T

A (1,1) predictor initialized to NT/NT mispredicts only at the first iter.

A Simple Example of (1,1) Predictor: Simulation - Cycle 0

```
     BNEZ R1, L1
     DADDIU R1, R0, #1
L1:  DSUBUI R3, R1, #-1
     BNEZ R3, L2
     …
L2:  …
```

b1

b2

Assumption: d alternates between 2 and 0

Initial Status

b1	NT 0	b1's last branch untaken
	NT 1	b1's last branch taken
b2	NT 0	b2's last branch untaken
	NT 1	b2's last branch taken

d=?	b1 pred.	b1 action	b1 new pred.	b2 pred.	b2 action	b2 new pred.
2	NT/NT	T		NT/NT	T	
0		NT			NT	
2		T			T	
0		NT			NT	

A (1,1) predictor initialized to NT/NT mispredicts only at the first iter.

A Simple Example of (1,1) Predictor: Simulation - Cycle 1

```
     BNEZ R1, L1
     DADDIU R1, R0, #1
L1:  DSUBUI R3, R1, #-1
     BNEZ R3, L2
     …
L2:  …
```

Assumption: d alternates between 2 and 0
X/Y: use X if last branch was not taken,
use Y if last branch was taken

Initial Status

b1	NT 0	b1's last branch untaken
	NT 1	b1's last branch taken
b2	NT 0	b2's last branch untaken
	NT 1	b2's last branch taken

d=?	b1 pred.	b1 action	b1 new pred.	b2 pred.	b2 action	b2 new pred.
2	NT/NT	T	T/NT	NT/**NT**	T	
0	T/NT	NT			NT	
2		T			T	
0		NT			NT	

A (1,1) predictor initialized to NT/NT mispredicts only at the first iter.

A Simple Example of (1,1) Predictor: Simulation - Cycle 2

```
     BNEZ R1, L1
     DADDIU R1, R0, #1
L1:  DSUBUI R3, R1, #-1
     BNEZ R3, L2
     …
L2:  …
```

Assumption: d alternates between 2 and 0
X/Y: use X if last branch was not taken,
use Y if last branch was taken

Initial Status

b1	NT 0	b1's last branch untaken
	NT 1	b1's last branch taken
b2	NT 0	b2's last branch untaken
	NT 1	b2's last branch taken

d=?	b1 pred.	b1 action	b1 new pred.	b2 pred.	b2 action	b2 new pred.
2	NT/NT	T	T/NT	NT/NT	T	NT/T
0	T/NT	NT		NT/T	NT	
2		T			T	
0		NT			NT	

A (1,1) predictor initialized to NT/NT mispredicts only at the first iter.

A Simple Example of (1,1) Predictor: Simulation - Cycle 3

```
     BNEZ R1, L1
     DADDIU R1, R0, #1
L1:  DSUBUI R3, R1, #-1
     BNEZ R3, L2
     …
L2:  …
```

Assumption: d alternates between 2 and 0
X/Y: use X if last branch was not taken,
use Y if last branch was taken

Initial Status

b1	NT 0	b1's last branch untaken
	NT 1	b1's last branch taken
b2	NT 0	b2's last branch untaken
	NT 1	b2's last branch taken

d=?	b1 pred.	b1 action	b1 new pred.	b2 pred.	b2 action	b2 new pred.
2	NT/NT	T	T/NT	NT/NT	T	NT/T
0	T/NT	NT	T/NT	NT/T	NT	
2	T/NT	T			T	
0		NT			NT	

A (1,1) predictor initialized to NT/NT mispredicts only at the first iter.

A Simple Example of (1,1) Predictor: Simulation - Cycle 4

```
     BNEZ R1, L1
     DADDIU R1, R0, #1
L1:  DSUBUI R3, R1, #-1
     BNEZ R3, L2
     …
L2:  …
```

Assumption: d alternates between 2 and 0
X/Y: use X if last branch was not taken,
use Y if last branch was taken

Initial Status

b1	NT 0	b1's last branch untaken
	NT 1	b1's last branch taken
b2	NT 0	b2's last branch untaken
	NT 1	b2's last branch taken

d=?	b1 pred.	b1 action	b1 new pred.	b2 pred.	b2 action	b2 new pred.
2	NT/NT	T	T/NT	NT/NT	T	NT/T
0	T/NT	NT	T/NT	NT/T	NT	NT/T
2	T/NT	T		NT/T	T	
0		NT			NT	

A (1,1) predictor initialized to NT/NT mispredicts only at the first iter.

A Simple Example of (1,1) Predictor: Simulation - Cycle 5

```
     BNEZ R1, L1
     DADDIU R1, R0, #1
L1:  DSUBUI R3, R1, #-1
     BNEZ R3, L2
     …
L2:  …
```

Assumption: d alternates between 2 and 0
X/Y: use X if last branch was not taken,
use Y if last branch was taken

Initial Status

b1	NT 0	b1's last branch untaken
	NT 1	b1's last branch taken
b2	NT 0	b2's last branch untaken
	NT 1	b2's last branch taken

d=?	b1 pred.	b1 action	b1 new pred.	b2 pred.	b2 action	b2 new pred.
2	NT/NT	T	T/NT	NT/NT	T	NT/T
0	T/NT	NT	T/NT	NT/T	NT	NT/T
2	T/NT	T	T/NT	NT/T	T	
0	T/NT	NT			NT	

A (1,1) predictor initialized to NT/NT mispredicts only at the first iter.

A Simple Example of (1,1) Predictor: Simulation - Cycle 6

```
     BNEZ R1, L1
     DADDIU R1, R0, #1
L1:  DSUBUI R3, R1, #-1
     BNEZ R3, L2
     …
L2:  …
```

Assumption: d alternates between 2 and 0
X/Y: use X if last branch was not taken,
use Y if last branch was taken

Initial Status

b1	NT 0	b1's last branch untaken
	NT 1	b1's last branch taken
b2	NT 0	b2's last branch untaken
	NT 1	b2's last branch taken

d=?	b1 pred.	b1 action	b1 new pred.	b2 pred.	b2 action	b2 new pred.
2	NT/NT	T	T/NT	NT/NT	T	NT/T
0	T/NT	NT	T/NT	NT/T	NT	NT/T
2	T/NT	T	T/NT	NT/T	T	NT/T
0	T/NT	NT		NT/T	NT	

A (1,1) predictor initialized to NT/NT mispredicts only at the first iter.

A Simple Example of (1,1) Predictor: Simulation - Cycle 7

```
     BNEZ R1, L1
     DADDIU R1, R0, #1
L1:  DSUBUI R3, R1, #-1
     BNEZ R3, L2
     …
L2:  …
```

Assumption: d alternates between 2 and 0
X/Y: use X if last branch was not taken, use Y if last branch was taken

Initial Status

b1	NT 0	b1's last branch untaken
	NT 1	b1's last branch taken
b2	NT 0	b2's last branch untaken
	NT 1	b2's last branch taken

d=?	b1 pred.	b1 action	b1 new pred.	b2 pred.	b2 action	b2 new pred.
2	NT/NT	T	T/NT	NT/NT	T	NT/T
0	T/NT	NT	T/NT	NT/T	NT	NT/T
2	T/NT	T	T/NT	NT/T	T	NT/T
0	T/NT	NT	T/NT	NT/T	NT	

A (1,1) predictor initialized to NT/NT mispredicts only at the first iter.

A Simple Example of (1,1) Predictor: Simulation - Cycle 8

```
     BNEZ R1, L1
     DADDIU R1, R0, #1
L1:  DSUBUI R3, R1, #-1
     BNEZ R3, L2
     …
L2:  …
```

Assumption: d alternates between 2 and 0
X/Y: use X if last branch was not taken, use Y if last branch was taken

Initial Status

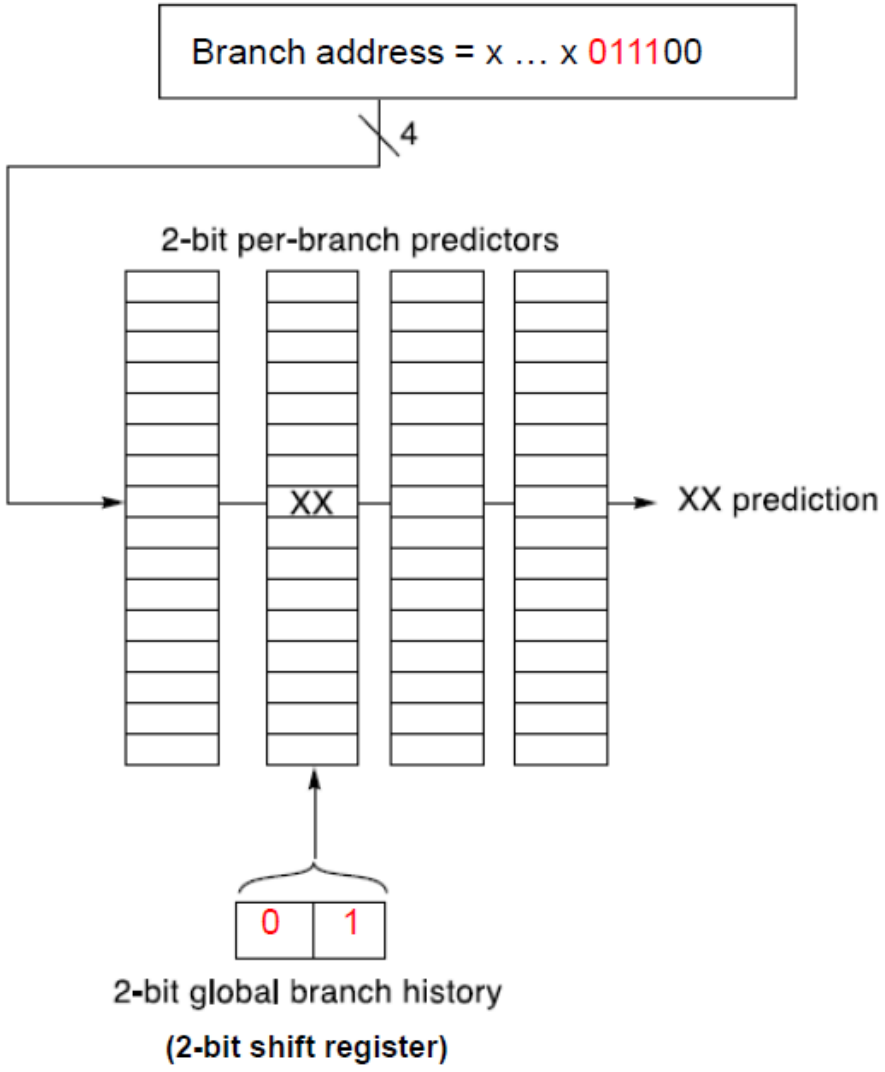
b1	NT 0	b1's last branch untaken
	NT 1	b1's last branch taken
b2	NT 0	b2's last branch untaken
	NT 1	b2's last branch taken

d=?	b1 pred.	b1 action	b1 new pred.	b2 pred.	b2 action	b2 new pred.
2	NT/NT	T	T/NT	NT/NT	T	NT/T
0	T/NT	NT	T/NT	NT/T	NT	NT/T
2	T/NT	T	T/NT	NT/T	T	NT/T
0	T/NT	NT	T/NT	NT/T	NT	NT/T

A (1,1) predictor initialized to NT/NT mispredicts only at the first iter.

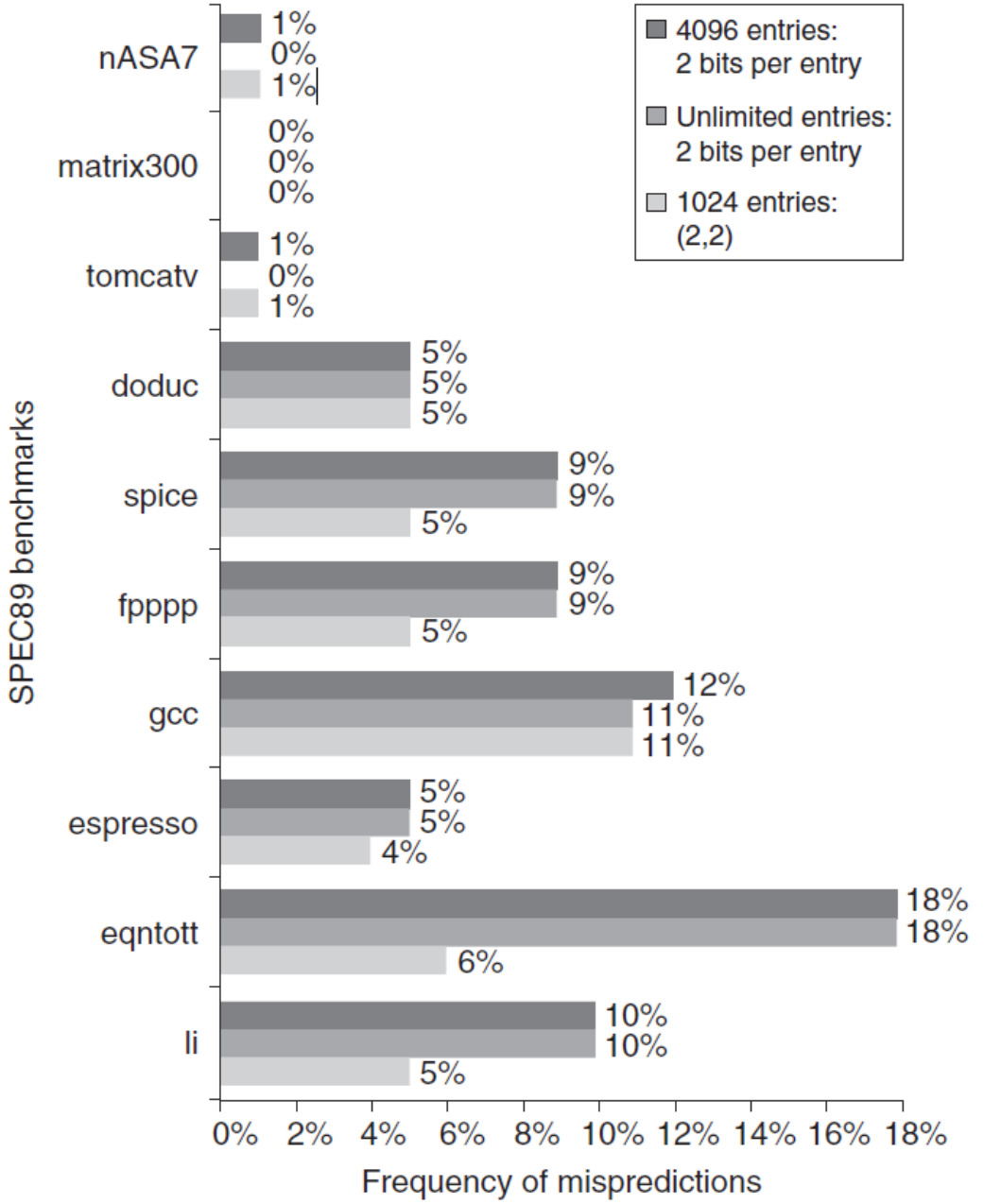
Example:
A 64-Entry (2,2) Branch Prediction Buffer

- Global history of the most recent m=2 branches is recorded in an 2-bit shift register where each bit records whether the branch was taken or not taken
- A concatenation of the low-order bits of the branch instruction address and the 2-bit global history is used to index the buffer and get the (n=2)-bit predictor
- Concept can be generalized to (m,n) predictor



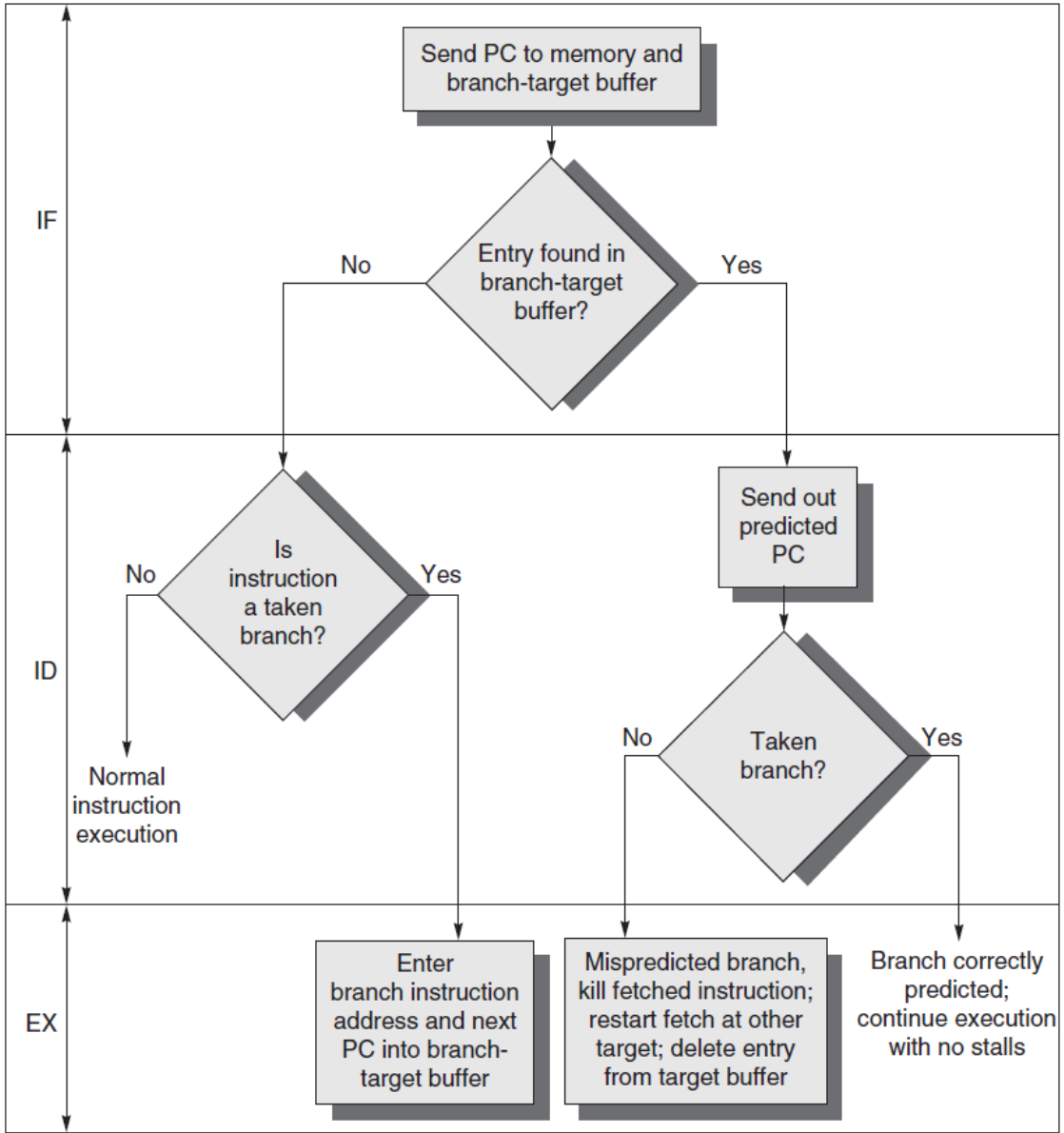
Compare 2-bit vs Correlating

- Three are in comparison:
  - 2-bit Predictor, (0,2)
  - (2,2) Correlating
  - 2-bit Predictor with unlimited entries.
- The first two have the same total number of bits in their BHT.



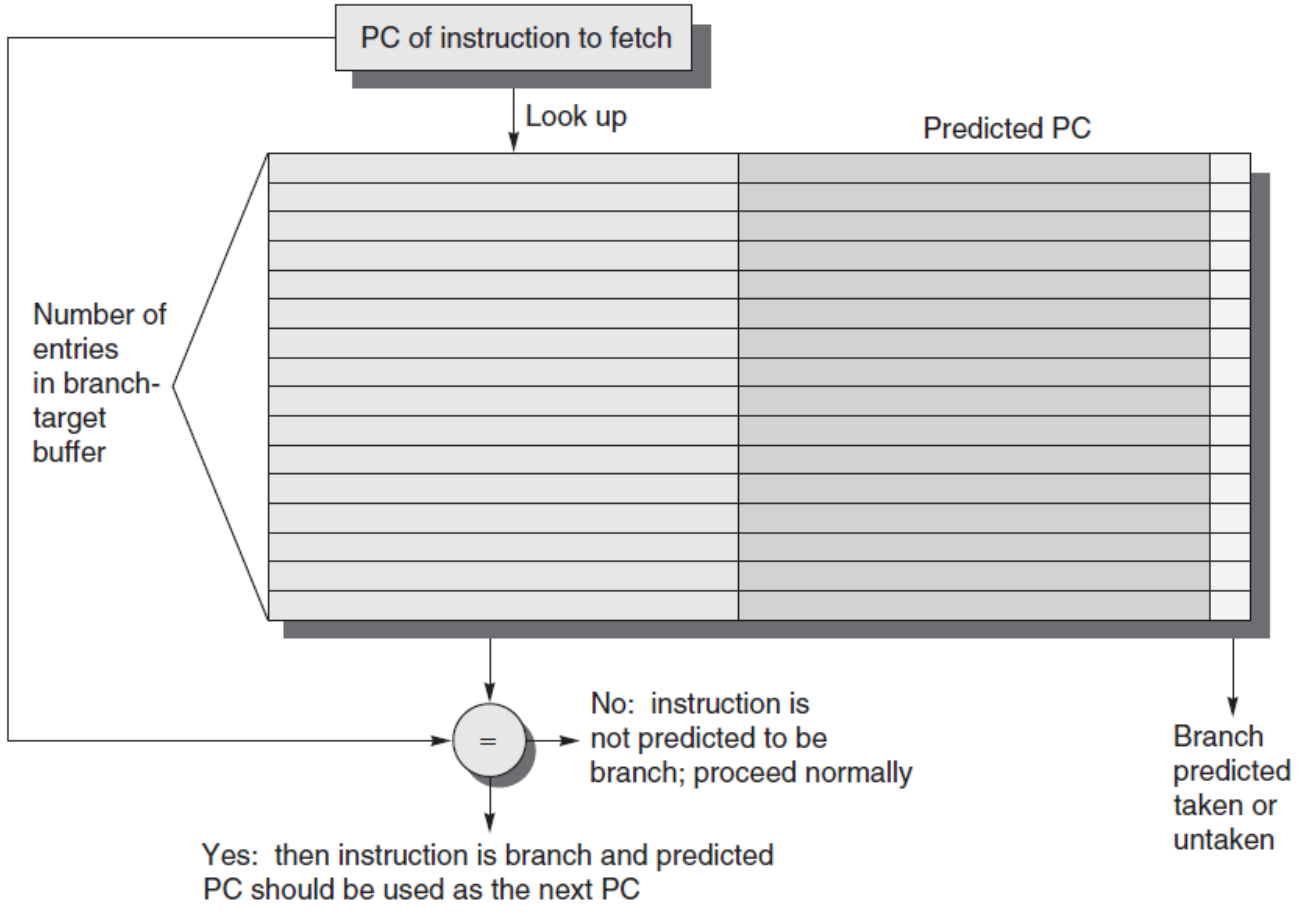
Predicting the Instruction Target Address: Branch-Target Buffer (or Branch-Target Cache)

- **Goal**: learn the **predicted instruction address at the** end of IF stage
  - one cycle earlier
    - at best, branch-prediction buffers know the next predicted instruction at the end of ID
- **No branch delay**
  - if entry found in buffer and prediction is correct
  - if entry not found and branch is not taken
- **Two cycle penalty**
  - if prediction is incorrect
  - if entry is not found and branch is taken



Predicting the Instruction Target Address: Branch-Target Buffer (or Branch-Target Cache)

- Only necessary to store the Predicted taken branches since an untaken branch follows the same strategy (to fetch the fall-through instruction) as a non-branch instruction
- But using a 2- bit predictor requires to store information also for untaken branches



Branch Target Buffer: Penalty Table

Instruction in buffer	Prediction	Actual Branch	Penalty Clock Cycles
yes	taken	taken	0
yes	taken	not taken	2
no		taken	2
no		not taken	0

- Assuming
  - –85% prediction accuracy, 90% buffer hit rate, 60% branches taken

- P(branch not in buffer, but taken) = 0.1 x 0.6 = 0.06
- P(branch in buffer but not taken) = 0.9 x 0.15 = 0.135
- Total branch penalty = (0.135 + 0.06) x 2 = 0.39
  - – the penalty for delayed-branch was about 0.5 clock cycles
  - – penalty is lower with better predictors (and bigger branch delays)

Example

Assume a machine that has a branch-target buffer with 8 entries. A branch in this machine has a penalty of 2 clock cycles if the branch is taken and the target instruction is not in the branch-target buffer, or if the branch is predicted as taken, the instruction is in the branch-target buffer, but the branch is actually not taken. In all other situations the branch penalty is zero. What is the total branch penalty in this machine, measured in clock cycles, if

- the branch prediction accuracy is 90%;
- 80% of the time the target instruction is in the buffer (80% hit rate in the buffer);
- 60% of the branches are actually taken.

Answer:

Probability of branch taken but not found in the buffer:

Percentage of taken branches * buffer miss rate = 60% * 20% = 0.12

Probability of branch found in buffer but predicted wrong:

Buffer hit rate * prediction miss rate = 80% * 10% = 0.08

Average branch penalty = (0.12 + 0.08) * 2 = 0.4 clock cycles