EE 156: Advanced Topics in Computer Architecture

Spring 2023
Tufts University

Instructor: Prof. Mark Hempstead
mark@ece.tufts.edu

Lecture 7: Branch Prediction and Advanced Branch Prediction

Outline

Unit 2: Microarchitecture and the Pipeline (3 weeks)

- *Instructions Introducing the RISC-V ISAs*
- *Basic Pipelining Review [Appendices A, C and K]*
- Hardware instruction-level parallelism (ILP) and Tomosulo's algorithm [Ch 3]
- **Advanced Branch Prediction**

Branch prediction

- We looked at two strategies for speeding up branches; assume-taken and assume-not-taken. Neither worked well.
- Now we'll look at branch prediction.
  - Hardware guesses branch outcome
  - Start fetching from guessed address
  - Flush the pipe on a mis-predict.

People branch-predict too

- Buy an engagement ring, predicting a "yes" answer
  - If you guessed wrong, then throw away the ring?
- Pick which way to drive to work based on what the traffic was like yesterday
  - If today's backups are different, then make excuses when you get to work

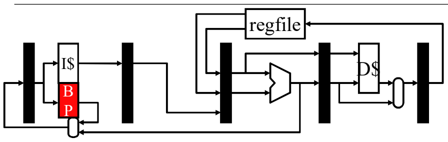
Branch prediction is hard

- Making predictions is hard, especially when they're about things that haven't happened yet.
  - Yogi Berra
- Corollary: branch prediction is hard

Branch Prediction Performance

- Parameters
  - **Branch: 20%**, load: 20%, store: 10%, other: 50%
  - 75% of branches are taken
- Dynamic branch prediction
  - Branches predicted with 95% accuracy
  - CPI = 1 + 20% * 5% * 2 = **1.02**
  - What is the CPI without branch prediction? (assume a cost of 2 cycles to resolve a taken branch and that 50 of branches are taken)
    - **CPI = 1 + 20% * 50% * 2 = 1.2**

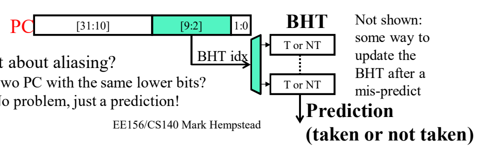
Dynamic Branch Prediction Components



- Step #1: is it a branch?
  - Easy after decode...
- Step #2: is the branch taken or not taken?
  - **Direction predictor** (applies to conditional branches only)
  - Predicts taken/not-taken
- Step #3: if the branch is taken, where does it go?
  - Branch **target predictor**
  - Easy after decode…

Branch Prediction

- **Learn from past, predict the future**
  - Record the past in a hardware structure
- **Direction predictor (DIRP)**
  - Map conditional-branch PC to taken/not-taken (T/N) decision
  - Individual conditional branches often biased or weakly biased
    - 90%+ one way or the other considered **"biased"**
    - Why? Loop back edges, checking for uncommon conditions
- **Branch history table (BHT)**: simplest predictor
  - PC indexes table of bits (0 = Not taken, 1 = Taken), no tags
  - Essentially: branch will go same way it went last time



Not shown: some way to update the BHT after a mis-predict

- What about aliasing?
  - Two PC with the same lower bits?
  - No problem, just a prediction!

Branch History Table (BHT)

- **Branch history table (BHT)**: simplest direction predictor
  - PC indexes table of bits (0 = N, 1 = T), no tags
  - Predicts the branch to go same way it went last time
  - Problem: **inner loop branch** below

```
for (i=0;i<100;i++)
  for (j=0;j<3;j++)
    // whatever
```

  - It will be wrong twice per inner loop; once each at the beginning and the end.
  - Branch predictor "changes its mind too quickly"

Time	State	Prediction	Outcome	Result?
1			T	
2			T	
3			T	
4			N	
5			T	
6			T	
7			T	
8			N	
9			T	
10			T	
11			T	
12			N	

Branch History Table (BHT)

- **Branch history table (BHT)**: simplest direction predictor
  - PC indexes table of bits (0 = N, 1 = T), no tags
  - Predicts the branch to go same way it went last time
  - Problem: **inner loop branch** below

```
for (i=0;i<100;i++)
    for (j=0;j<3;j++)
        // whatever
```

    - It will be wrong twice per inner loop; once each at the beginning and the end.
    - Branch predictor "changes its mind too quickly"

Time	State	Prediction	Outcome	Result?
1	N	N	T	Wrong
2	T			
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				

Branch History Table (BHT)

- **Branch history table (BHT)**: simplest direction predictor
  - PC indexes table of bits (0 = N, 1 = T), no tags
  - Predicts the branch to go same way it went last time
  - Problem: **inner loop branch** below

```
for (i=0;i<100;i++)
    for (j=0;j<3;j++)
        // whatever
```

    - It will be wrong twice per inner loop; once each at the beginning and the end.
    - Branch predictor "changes its mind too quickly"

Time	State	Prediction	Outcome	Result?
1	N	N	T	Wrong
2	T	T	T	Correct
3	T			
4				
5				
6				
7				
8				
9				
10				
11				
12				

Branch History Table (BHT)

- **Branch history table (BHT)**: simplest direction predictor
  - PC indexes table of bits (0 = N, 1 = T), no tags
  - Predicts the branch to go same way it went last time
  - Problem: **inner loop branch** below

```
for (i=0;i<100;i++)
    for (j=0;j<3;j++)
        // whatever
```

    - It will be wrong twice per inner loop; once each at the beginning and the end.
    - Branch predictor "changes its mind too quickly"

Time	State	Prediction	Outcome	Result?
1	N	N	T	Wrong
2	T	T	T	Correct
3	T	T	T	Correct
4	T			
5				
6				
7				
8				
9				
10				
11				
12				

Branch History Table (BHT)

- **Branch history table (BHT)**: simplest direction predictor
  - PC indexes table of bits (0 = N, 1 = T), no tags
  - Predicts the branch to go same way it went last time
  - Problem: **inner loop branch** below

```
for (i=0;i<100;i++)
    for (j=0;j<3;j++)
        // whatever
```

  - It will be wrong twice per inner loop; once each at the beginning and the end.
  - Branch predictor "changes its mind too quickly"

Time	State	Prediction	Outcome	Result?
1	N	N	T	Wrong
2	T	T	T	Correct
3	T	T	T	Correct
4	T	T	N	Wrong
5	N			
6				
7				
8				
9				
10				
11				
12				

Branch History Table (BHT)

- **Branch history table (BHT)**: simplest direction predictor
  - PC indexes table of bits (0 = N, 1 = T), no tags
  - Predicts the branch to go same way it went last time
  - Problem: **inner loop branch** below

```
for (i=0;i<100;i++)
    for (j=0;j<3;j++)
        // whatever
```

  - It will be wrong twice per inner loop; once each at the beginning and the end.
  - Branch predictor "changes its mind too quickly"

Time	State	Prediction	Outcome	Result?
1	N	N	T	Wrong
2	T	T	T	Correct
3	T	T	T	Correct
4	T	T	N	Wrong
5	N	N	T	Wrong
6	T			
7				
8				
9				
10				
11				
12				

Branch History Table (BHT)

- **Branch history table (BHT)**: simplest direction predictor
  - PC indexes table of bits (0 = N, 1 = T), no tags
  - Predicts the branch to go same way it went last time
  - Problem: **inner loop branch** below

```
for (i=0;i<100;i++)
    for (j=0;j<3;j++)
        // whatever
```

  - It will be wrong twice per inner loop; once each at the beginning and the end.
  - Branch predictor "changes its mind too quickly"

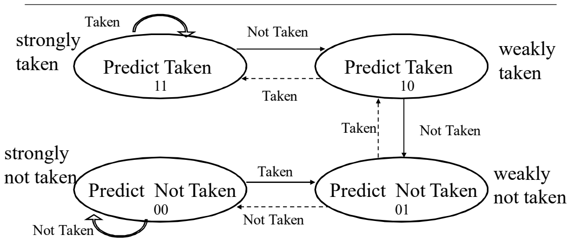
Time	State	Prediction	Outcome	Result?
1	N	N	T	Wrong
2	T	T	T	Correct
3	T	T	T	Correct
4	T	T	N	Wrong
5	N	N	T	Wrong
6	T	T	T	Correct
7	T	T	T	Correct
8	T	T	N	Wrong
9	N	N	T	Wrong
10	T	T	T	Correct
11	T	T	T	Correct
12	T	T	N	Wrong

Two-Bit Saturating Counters (2bc)

- **Two-bit saturating counters (2bc)** [Smith 1981]
  - Replace each single-bit predictor with
    - (0,1,2,3) = (N,n,t,T)
  - Adds “hysteresis”
    - Force predictor to mis-predict twice before “changing its mind”
  - One mispredict each loop execution (rather than two)
    - + Improves our inner-loop problem.

Time	State	Prediction	Outcome	Result?
1	N	N	T	Wrong
2	n			
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				

Two-bit Saturating Counters



- 2-bit FSMs mean prediction must miss twice before change
- N-bit predictors are possible, but after 2-bits not much benefit

Two-Bit Saturating Counters (2bc)

- **Two-bit saturating counters (2bc)** [Smith 1981]
  - Replace each single-bit predictor with
    - (0,1,2,3) = (N,n,t,T)
  - Adds “hysteresis”
    - Force predictor to mis-predict twice before “changing its mind”
  - One mispredict each loop execution (rather than two)
    - + Improves our inner-loop problem.

Time	State	Prediction	Outcome	Result?
1	N	N	T	Wrong
2	n	N	T	Wrong
3	t			
4				
5				
6				
7				
8				
9				
10				
11				
12				

Two-Bit Saturating Counters (2bc)

- **Two-bit saturating counters (2bc)** [Smith 1981]
  - Replace each single-bit predictor with
    - (0,1,2,3) = (N,n,t,T)
  - Adds "hysteresis"
    - Force predictor to mis-predict twice before "changing its mind"
  - One mispredict each loop execution (rather than two)
    - + Improves our inner-loop problem.

Time	State	Prediction	Outcome	Result?
1	N	N	T	Wrong
2	n	N	T	Wrong
3	t	T	T	Correct
4	T			
5				
6				
7				
8				
9				
10				
11				
12				

Two-Bit Saturating Counters (2bc)

- **Two-bit saturating counters (2bc)** [Smith 1981]
  - Replace each single-bit predictor with
    - (0,1,2,3) = (N,n,t,T)
  - Adds "hysteresis"
    - Force predictor to mis-predict twice before "changing its mind"
  - One mispredict each loop execution (rather than two)
    - + Improves our inner-loop problem.

Time	State	Prediction	Outcome	Result?
1	N	N	T	Wrong
2	n	N	T	Wrong
3	t	T	T	Correct
4	T	T	N	Wrong
5	t			
6				
7				
8				
9				
10				
11				
12				

Two-Bit Saturating Counters (2bc)

- **Two-bit saturating counters (2bc)** [Smith 1981]
  - Replace each single-bit predictor with
    - (0,1,2,3) = (N,n,t,T)
  - Adds "hysteresis"
    - Force predictor to mis-predict twice before "changing its mind"
  - One mispredict each loop execution (rather than two)
    - + Improves our inner-loop problem.

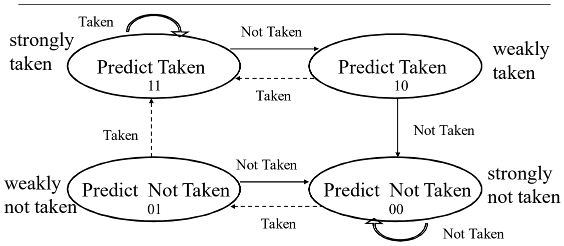
Time	State	Prediction	Outcome	Result?
1	N	N	T	Wrong
2	n	N	T	Wrong
3	t	T	T	Correct
4	T	T	N	Wrong
5	t	T	T	Correct
6	T			
7				
8				
9				
10				
11				
12				

Two-Bit Saturating Counters (2bc)

- **Two-bit saturating counters (2bc)** [Smith 1981]
  - Replace each single-bit predictor with
    - (0,1,2,3) = (N,n,t,T)
  - Adds “hysteresis”
    - Force predictor to mis-predict twice before “changing its mind”
  - One mispredict each loop execution (rather than two)
    - + Improves our inner-loop problem.

Time	State	Prediction	Outcome	Result?
1	N	N	T	Wrong
2	n	N	T	Wrong
3	t	T	T	Correct
4	T	T	N	Wrong
5	t	T	T	Correct
6	T	T	T	Correct
7	T	T	T	Correct
8	T	T	N	Wrong
9	t	T	T	Correct
10	T	T	T	Correct
11	T	T	T	Correct
12	T	T	N	Wrong

Two-bit Saturating Counters, take 2



- Slightly different version works better for some benchmarks.

Intuition on how well this works

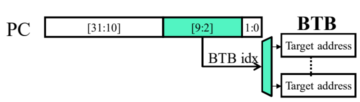
- Simple predictors work well:
  - on branches that repeat the same behavior in streaks.
  - They work better when the streaks are longer
- They don’t work well:
  - for random or data-dependent branches.
  - Data dependence can be common in some applications, but it’s evil for branch prediction.

Branch Target Buffer (BTB)

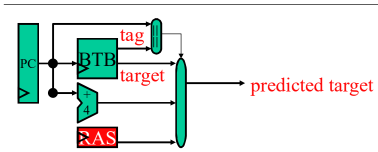
- It's little use predicting taken/not early in ID if you don't know the target address until late in EX.
  - So we record the past branch targets in a hardware structure
- Branch target buffer (BTB):
  - "Last time the branch X was taken, it went to address Y"
    - "So, in the future, if address X is fetched, fetch address Y next"
- Works well because most control insns use **direct targets**
  - Target encoded in insn itself → same "taken" target every time
- What about **indirect targets**?
  - Target held in a register → can be different each time
  - Two indirect call idioms
    - Dynamically linked functions (DLLs): target always the same
    - Dynamically dispatched (virtual) functions: hard but uncommon
  - Also two indirect unconditional jump idioms
    - Switches: hard but uncommon
    - Function returns: hard and common but…

Implementation

- A small RAM: address = PC, data = target-PC
- Access at Fetch *in parallel* with instruction memory
  - predicted-target = BTB[hash(PC)]
- Updated whenever target != predicted-target
  - BTB[hash(PC)] = correct-target
- Very similar to BHT, but it's a full address width and not just one taken/not bit.
- Aliasing? No problem, this is only a prediction



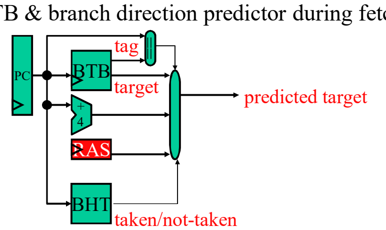
Return Address Stack (RAS)



- **Return address stack (RAS)**
  - Call instruction? RAS[TopOfStack++] = PC+4
  - Return instruction? Predicted-target = RAS[--TopOfStack]
  - Q: how can you tell if an insn is a call/return before decoding it?
    - Accessing RAS on every instruction would waste power
  - Answer:
    - another predictor (or put them in BTB marked as "return")
    - Or, **pre-decode bits** in insn mem, written when first executed

Putting It All Together

- BTB & branch direction predictor during fetch



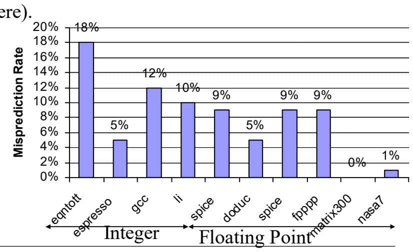
- *If* branch prediction correct, then no penalty for branches!

Branch Prediction Performance

- Dynamic branch prediction
  - 20% of instruction branches
  - Simple predictor: branches predicted with 75% accuracy
    - CPI = 1 + (20% * **25%** * 2) = **1.1**
  - More advanced predictor: 95% accuracy
    - CPI = 1 + (20% * **5%** * 2) = **1.02**
- Branch mis-predictions still a big problem though
  - Pipelines are long: typical mis-prediction penalty is 10+ cycles
  - For cores that do more per cycle, predictions most costly (later)
- Many other methods for building a better branch predictor including correlating predictors, global history tables(Chapter 3)

BHT Accuracy

- Mispredict because either:
  - Wrong guess for that branch
  - Got branch history of wrong branch due to aliasing (4K-entry table shown here).



Another way: Prediction accuracy between 99% and 82%

ADVANCED BRANCH PREDICTION (3.3)

Many more effective and complex branch prediction techniques exist

- Correlating branch predictors (sec 3.3)
  - Gshare correlating predictor (paper and lab)
- Tournament predictors (sec 3.3)
- Tagged Hybrid predictors (sec 3.3)
- Core i7 branch predictors (sec 3.3)
- Predictors base on machine learning
  - Perceptron branch prediction (see paper)

Correlating Predictors

- 2-bit scheme only looks at branch's *own* history to predict its behavior
- What if we use other branches to predict it as well?

```
if (aa==2)aa=0;     // Branch #1
if (bb==2)bb=0;     // Branch #2
if (aa!=bb){..}     // Branch #3
```

- Does branch #3 depends on outcome of #1 and #2?
  - Yes. If #1 & #2 are both taken, then #3 will not be taken.
  - Sometimes the past can predict the future with certainty!
  - This is called "global history."

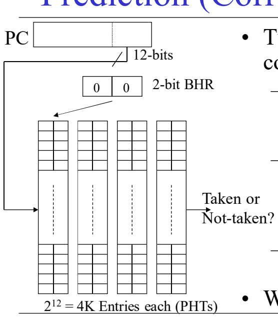
Another example of global history

- Consider our inner loop again
  - The branch pattern is (T,T,T,N) repeating
- Consider the following global rule:
  - If the last branches were “T,T,T” then predict not taken
  - Else predict taken.
  - Works perfectly – the trick is having the predictor figure out that rule.
- Let’s make a predictor that looks at both the history at this branch, as well as recent outcomes of all branches (including this one).

Branch History Register

- Simple shift register
  - Shift in branch outcomes as they occur
  - 1 => branch was taken
  - 0 => branch was not-taken
  - k-bit BHR => 2^k patterns
  - Now we know the outcome of the last *k* branches.
  - Use these patterns to address into the Pattern History Table. Effectively the PHT is a separate BHT for each global pattern.

Generic Adaptive Branch Prediction (Correlating Predictor)



- Two-level BP requires two main components
  - Branch history register (BHR): recent outcomes of branches (last **k** branches encountered)
  - Pattern History Table (PHT): branch behavior for recent occurrences of the specific pattern of these **k** branches
  - In effect, we concatenate BHR with Branch PC bits
- We’ve drawn a (2,2) predictor

Pattern History Table

- Has 2^k entries
- Usually uses a 2-bit counter for the prediction
- BHR is used to address the PHT. I.e., a (k,2) PHT is an array of 2^k BHTs, each of which has a 2-bit counter.

(2,1) correlated Predictor

- **Correlated (two-level) predictor** [Patt 1991]
  - Exploits observation that branch outcomes are correlated
  - Maintains separate prediction per (PBHR, PC) pairs
    - **Branch history register (BHR)**: recent branch outcomes
  - Simple working example: assume program has one branch
    - BHT: one 1-bit DIRP entry
    - BHT+**2BHR**: 2^2 = **4** 1-bit DIRP entries

Time	BHR	PHT NN	PHT NT	PHT TN	PHT TT	Prediction	Outcome	Result?
1	NN	N	N	N	N		T	
2							T	
3							T	
4							N	
5							T	
6							T	
7							T	
8							N	
9							T	
10							T	
11							T	
12							N	

(2,1) correlated Predictor

- **Correlated (two-level) predictor** [Patt 1991]
  - Exploits observation that branch outcomes are correlated
  - Maintains separate prediction per (PC, BHR) pairs
    - **Branch history register (BHR)**: recent branch outcomes
  - Simple working example: assume program has one branch
    - BHT: one 1-bit DIRP entry
    - BHT+**2BHR**: 2^2 = **4** 1-bit DIRP entries

Time	BHR	PHT NN	PHT NT	PHT TN	PHT TT	Prediction	Outcome	Result?
1	NN	N	N	N	N	N	T	Wrong
2	NT	T	N	N	N			
3								
4								
5								
6								
7								
8								
9								
10								
11								
12								

(2,1) correlated Predictor

- **Correlated (two-level) predictor** [Patt 1991]
  - Exploits observation that branch outcomes are correlated
  - Maintains separate prediction per (PC, BHR) pairs
    - **Branch history register (BHR)**: recent branch outcomes
  - Simple working example: assume program has one branch
    - BHT: one 1-bit DIRP entry
    - BHT+**2BHR**: 2^2 = **4** 1-bit DIRP entries

Time	BHR	PHT NN	PHT NT	PHT TN	PHT TT	Prediction	Outcome	Result?
1	NN	**N**	N	N	N	**N**	**T**	Wrong
2	NT	T	**N**	N	N	**N**	**T**	Wrong
3	TT	T	T	N	N			
4								
5								
6								
7								
8								
9								
10								
11								
12								

(2,1) correlated Predictor

- **Correlated (two-level) predictor** [Patt 1991]
  - Exploits observation that branch outcomes are correlated
  - Maintains separate prediction per (PC, BHR) pairs
    - **Branch history register (BHR)**: recent branch outcomes
  - Simple working example: assume program has one branch
    - BHT: one 1-bit DIRP entry
    - BHT+**2BHR**: 2^2 = **4** 1-bit DIRP entries

Time	BHR	PHT NN	PHT NT	PHT TN	PHT TT	Prediction	Outcome	Result?
1	NN	**N**	N	N	N	**N**	**T**	Wrong
2	NT	T	**N**	N	N	**N**	**T**	Wrong
3	TT	T	T	N	**N**	**N**	**T**	Wrong
4	TT	T	T	N	T			
5								
6								
7								
8								
9								
10								
11								
12								

(2,1) correlated Predictor

- **Correlated (two-level) predictor** [Patt 1991]
  - Exploits observation that branch outcomes are correlated
  - Maintains separate prediction per (PC, BHR) pairs
    - **Branch history register (BHR)**: recent branch outcomes
  - Simple working example: assume program has one branch
    - BHT: one 1-bit DIRP entry
    - BHT+**2BHR**: 2^2 = **4** 1-bit DIRP entries

Time	BHR	PHT NN	PHT NT	PHT TN	PHT TT	Prediction	Outcome	Result?
1	NN	**N**	N	N	N	**N**	**T**	Wrong
2	NT	T	**N**	N	N	**N**	**T**	Wrong
3	TT	T	T	N	**N**	**N**	**T**	Wrong
4	TT	T	T	N	**T**	**T**	**N**	Wrong
5	TN	T	T	N	N			
6								
7								
8								
9								
10								
11								
12								

(2,1) correlated Predictor

- **Correlated (two-level) predictor** [Patt 1991]
  - Exploits observation that branch outcomes are correlated
  - Maintains separate prediction per (PC, BHR) pairs
    - **Branch history register (BHR)**: recent branch outcomes
  - Simple working example: assume program has one branch
    - BHT: one 1-bit DIRP entry
    - BHT+**2BHR**: 2^2 = **4** 1-bit DIRP entries

Time	BHR	PHT NN	PHT NT	PHT TN	PHT TT	Prediction	Outcome	Result?
1	NN	N	N	N	N	N	T	Wrong
2	NT	T	N	N	N	N	T	Wrong
3	TT	T	T	N	N	N	T	Wrong
4	TT	T	T	N	T	T	N	Wrong
5	TN	T	T	N	N	N	T	Wrong
6	NT	T	T	T	N			
7								
8								
9								
10								
11								
12								

43

(2,1) correlated Predictor

- **Correlated (two-level) predictor** [Patt 1991]
  - Exploits observation that branch outcomes are correlated
  - Maintains separate prediction per (PC, BHR) pairs
    - **Branch history register (BHR)**: recent branch outcomes
  - Simple working example: assume program has one branch
    - BHT: one 1-bit DIRP entry
    - BHT+**2BHR**: 2^2 = **4** 1-bit DIRP entries

Time	BHR	PHT NN	PHT NT	PHT TN	PHT TT	Prediction	Outcome	Result?
1	NN	N	N	N	N	N	T	Wrong
2	NT	T	N	N	N	N	T	Wrong
3	TT	T	T	N	N	N	T	Wrong
4	TT	T	T	N	T	T	N	Wrong
5	TN	T	T	N	N	N	T	Wrong
6	NT	T	T	T	N	T	T	Correct
7	TT	T	T	T	N			
8								
9								
10								
11								
12								

44

(2,1) correlated Predictor

- **Correlated (two-level) predictor** [Patt 1991]
  - Exploits observation that branch outcomes are correlated
  - Maintains separate prediction per (PC, BHR) pairs
    - **Branch history register (BHR)**: recent branch outcomes
  - Simple working example: assume program has one branch
    - BHT: one 1-bit DIRP entry
    - BHT+**2BHR**: 2^2 = **4** 1-bit DIRP entries

Time	BHR	PHT NN	PHT NT	PHT TN	PHT TT	Prediction	Outcome	Result?
1	NN	N	N	N	N	N	T	Wrong
2	NT	T	N	N	N	N	T	Wrong
3	TT	T	T	N	N	N	T	Wrong
4	TT	T	T	N	T	T	N	Wrong
5	TN	T	T	N	N	N	T	Wrong
6	NT	T	T	T	N	T	T	Correct
7	TT	T	T	T	N	N	T	Wrong
8	TT	T	T	T	T			
9								
10								
11								
12								

45

(2,1) correlated Predictor

- **Correlated (two-level) predictor** [Patt 1991]
  - Exploits observation that branch outcomes are correlated
  - Maintains separate prediction per (PC, BHR) pairs
    - **Branch history register (BHR)**: recent branch outcomes
  - Simple working example: assume program has one branch
    - BHT: one 1-bit DIRP entry
    - BHT+**2BHR**: 2^2 = **4** 1-bit DIRP entries

Time	BHR	PHT NN	PHT NT	PHT TN	PHT TT	Prediction	Outcome	Result?
1	NN	N	N	N	N	N	T	Wrong
2	NT	T	N	N	N	N	T	Wrong
3	TT	T	T	N	N	N	T	Wrong
4	TT	T	T	N	T	T	N	Wrong
5	TN	T	T	N	N	N	T	Wrong
6	NT	T	T	T	N	T	T	Correct
7	TT	T	T	T	N	N	T	Wrong
8	TT	T	T	T	T	T	N	Wrong
9	TN	T	T	T	N			
10								
11								
12								

46

(2,1) correlated Predictor

- **Correlated (two-level) predictor** [Patt 1991]
  - Exploits observation that branch outcomes are correlated
  - Maintains separate prediction per (PC, BHR) pairs
    - **Branch history register (BHR)**: recent branch outcomes
  - Simple working example: assume program has one branch
    - BHT: one 1-bit DIRP entry
    - BHT+**2BHR**: 2^2 = **4** 1-bit DIRP entries

Time	BHR	PHT NN	PHT NT	PHT TN	PHT TT	Prediction	Outcome	Result?
1	NN	N	N	N	N	N	T	Wrong
2	NT	T	N	N	N	N	T	Wrong
3	TT	T	T	N	N	N	T	Wrong
4	TT	T	T	N	T	T	N	Wrong
5	TN	T	T	N	N	N	T	Wrong
6	NT	T	T	T	N	T	T	Correct
7	TT	T	T	T	N	N	T	Wrong
8	TT	T	T	T	T	T	N	Wrong
9	TN	T	T	T	N	T	T	Correct
10	NT	T	T	T	N			
11								
12								

47

(2,1) correlated Predictor

- **Correlated (two-level) predictor** [Patt 1991]
  - Exploits observation that branch outcomes are correlated
  - Maintains separate prediction per (PC, BHR) pairs
    - **Branch history register (BHR)**: recent branch outcomes
  - Simple working example: assume program has one branch
    - BHT: one 1-bit DIRP entry
    - BHT+**2BHR**: 2^2 = **4** 1-bit DIRP entries

Time	BHR	PHT NN	PHT NT	PHT TN	PHT TT	Prediction	Outcome	Result?
1	NN	N	N	N	N	N	T	Wrong
2	NT	T	N	N	N	N	T	Wrong
3	TT	T	T	N	N	N	T	Wrong
4	TT	T	T	N	T	T	N	Wrong
5	TN	T	T	N	N	N	T	Wrong
6	NT	T	T	T	N	T	T	Correct
7	TT	T	T	T	N	N	T	Wrong
8	TT	T	T	T	T	T	N	Wrong
9	TN	T	T	T	N	T	T	Correct
10	NT	T	T	T	N	T	T	Correct
11	TT	T	T	T	N			
12								

48

(2,1) correlated Predictor

- **Correlated (two-level) predictor** [Patt 1991]
  - Exploits observation that branch outcomes are correlated
  - Maintains separate prediction per (PC, BHR) pairs
    - **Branch history register (BHR)**: recent branch outcomes
  - Simple working example: assume program has one branch
    - BHT: one 1-bit DIRP entry
    - BHT+**2BHR**: 2^2 = **4** 1-bit DIRP entries

Time	BHR	PHT NN	PHT NT	PHT TN	PHT TT	Prediction	Outcome	Result?
1	NN	N	N	N	N	N	T	Wrong
2	NT	T	N	N	N	N	T	Wrong
3	TT	T	T	N	N	N	T	Wrong
4	TT	T	T	N	T	T	N	Wrong
5	TN	T	T	N	N	N	T	Wrong
6	NT	T	T	T	N	T	T	Correct
7	TT	T	T	T	N	N	T	Wrong
8	TT	T	T	T	T	T	N	Wrong
9	TN	T	T	T	N	T	T	Correct
10	NT	T	T	T	N	T	T	Correct
11	TT	T	T	T	N	N	T	Wrong
12	TT	T	T	T	T			

49

(2,1) correlated Predictor

- **What went wrong?**
  - Look at the grey squares; they log our mistakes.
  - For NN, NT, TN we only made one mistake. But TT kept making mistakes. Why?
    - We have both T,T,T and also T,T,N.
    - Every time we went from one to the other we got the wrong answer and had to re-train our predictor.
  - How do we fix this simply?
    - Go to a (3,1) predictor.

Time	BHR	PHT NN	PHT NT	PHT TN	PHT TT	Prediction	Outcome	Result?
1	NN	N	N	N	N	N	T	Wrong
2	NT	T	N	N	N	N	T	Wrong
3	TT	T	T	N	N	N	T	Wrong
4	TT	T	T	N	T	T	N	Wrong
5	TN	T	T	N	N	N	T	Wrong
6	NT	T	T	T	N	T	T	Correct
7	TT	T	T	T	N	N	T	Wrong
8	TT	T	T	T	T	T	N	Wrong
9	TN	T	T	T	N	T	T	Correct
10	NT	T	T	T	N	T	T	Correct
11	TT	T	T	T	N	N	T	Wrong
12	TT	T	T	T	T	T	N	Wrong

50

(3,1) correlated predictor

- 3-bit BHR
- 2^3 DIRP entries per pattern
- One-bit BHT (not a 2-bit saturating counter)

Time	BHR	PHT NNN	PHT NNT	PHT NTN	PHT NTT	PHT TNN	PHT TNT	PHT TTN	PHT TTT	Prediction	Outcome	Result?
1											T	
2											T	
3											T	
4											N	
5											T	
6											T	
7											T	
8											N	
9											T	
10											T	
11											T	
12											N	

51

(3,1) correlated predictor

- 3-bit BHR
- 2^3 DIRP entries per pattern
- One-bit BHT (not a 2-bit saturating counter)

Time	BHR	PHT NNN	NNT	NTN	NTT	TNN	TNT	TTN	TTT	Prediction	Outcome	Result?
1	NNN	**N**	N	N	N	N	N	N	N	N	T	Wrong
2	NNT	T	N	N	N	N	N	N	N			
3												
4												
5												
6												
7												
8												
9												
10												
11												
12												

52

(3,1) correlated predictor

- 3-bit BHR
- 2^3 DIRP entries per pattern
- One-bit BHT (not a 2-bit saturating counter)

Time	BHR	PHT NNN	NNT	NTN	NTT	TNN	TNT	TTN	TTT	Prediction	Outcome	Result?
1	NNN	**N**	N	N	N	N	N	N	N	N	T	Wrong
2	NNT	T	**N**	N	N	N	N	N	N	N	T	Wrong
3	NTT	T	T	N	N	N	N	N	N			
4												
5												
6												
7												
8												
9												
10												
11												
12												

53

(3,1) correlated predictor

- 3-bit BHR
- 2^3 DIRP entries per pattern
- One-bit BHT (not a 2-bit saturating counter)

Time	BHR	PHT NNN	NNT	NTN	NTT	TNN	TNT	TTN	TTT	Prediction	Outcome	Result?
1	NNN	**N**	N	N	N	N	N	N	N	N	T	Wrong
2	NNT	T	**N**	N	N	N	N	N	N	N	T	Wrong
3	NTT	T	T	N	**N**	N	N	N	N	N	T	Wrong
4	TTT	T	T	N	T	N	N	N	N			
5												
6												
7												
8												
9												
10												
11												
12												

54

(3,1) correlated predictor

- 3-bit BHR
- 2^3 DIRP entries per pattern
- One-bit BHT (not a 2-bit saturating counter)

Time	BHR	PHT: NNN	NNT	NTN	NTT	TNN	TNT	TTN	TTT	Prediction	Outcome	Result?
1	NNN	N	N	N	N	N	N	N	N	N	T	Wrong
2	NNT	T	N	N	N	N	N	N	N	N	T	Wrong
3	NTT	T	T	N	N	N	N	N	N	N	T	Wrong
4	TTT	T	T	N	T	N	N	N	N	N	N	Correct
5	TTN	T	T	N	T	N	N	N	N			
6												
7												
8												
9												
10												
11												
12												

(3,1) correlated predictor

- 3-bit BHR
- 2^3 DIRP entries per pattern
- One-bit BHT (not a 2-bit saturating counter)

Time	BHR	PHT: NNN	NNT	NTN	NTT	TNN	TNT	TTN	TTT	Prediction	Outcome	Result?
1	NNN	N	N	N	N	N	N	N	N	N	T	Wrong
2	NNT	T	N	N	N	N	N	N	N	N	T	Wrong
3	NTT	T	T	N	N	N	N	N	N	N	T	Wrong
4	TTT	T	T	N	T	N	N	N	N	N	N	Correct
5	TTN	T	T	N	T	N	N	N	N	N	T	Wrong
6	TNT	T	T	N	T	N	N	T	N			
7												
8												
9												
10												
11												
12												

(3,1) correlated predictor

- 3-bit BHR
- 2^3 DIRP entries per pattern
- One-bit BHT (not a 2-bit saturating counter)

Time	BHR	PHT: NNN	NNT	NTN	NTT	TNN	TNT	TTN	TTT	Prediction	Outcome	Result?
1	NNN	N	N	N	N	N	N	N	N	N	T	Wrong
2	NNT	T	N	N	N	N	N	N	N	N	T	Wrong
3	NTT	T	T	N	N	N	N	N	N	N	T	Wrong
4	TTT	T	T	N	T	N	N	N	N	N	N	Correct
5	TTN	T	T	N	T	N	N	N	N	N	T	Wrong
6	TNT	T	T	N	T	N	N	T	N	N	T	Wrong
7	NTT	T	T	N	T	N	T	T	N			
8												
9												
10												
11												
12												

(3,1) correlated predictor

- 3-bit BHR
- 2^3 DIRP entries per pattern
- One-bit BHT (not a 2-bit saturating counter)

Time	BHR	PHT NNN	NNT	NTN	NTT	TNN	TNT	TTN	TTT	Prediction	Outcome	Result?
1	NNN	N	N	N	N	N	N	N	N	N	T	Wrong
2	NNT	T	N	N	N	N	N	N	N	N	T	Wrong
3	NTT	T	T	N	N	N	N	N	N	N	T	Wrong
4	TTT	T	T	N	T	N	N	N	N	N	N	Correct
5	TTN	T	T	N	T	N	N	N	N	N	T	Wrong
6	TNT	T	T	N	T	N	N	T	N	N	T	Wrong
7	NTT	T	T	N	T	N	T	T	N	T	T	Correct
8	TTT	T	T	N	T	N	T	T	N			
9												
10												
11												
12												

(3,1) correlated predictor

- With 3 bits of history, the history is now enough to fully predict the branch result.
- Any column has only one grey square; once we train the predictor for a given history it never makes another mistake.

Time	BHR	PHT NNN	NNT	NTN	NTT	TNN	TNT	TTN	TTT	Prediction	Outcome	Result?
1	NNN	N	N	N	N	N	N	N	N	N	T	Wrong
2	NNT	T	N	N	N	N	N	N	N	N	T	Wrong
3	NTT	T	T	N	N	N	N	N	N	N	T	Wrong
4	TTT	T	T	N	T	N	N	N	N	N	N	Correct
5	TTN	T	T	N	T	N	N	N	N	N	T	Wrong
6	TNT	T	T	N	T	N	N	T	N	N	T	Wrong
7	NTT	T	T	N	T	N	T	T	N	T	T	Correct
8	TTT	T	T	N	T	N	T	T	N	N	N	Correct
9	TTN	T	T	N	T	N	T	T	N	T	T	Correct
10	TNT	T	T	N	T	N	T	T	N	T	T	Correct
11	NTT	T	T	N	T	N	T	T	N	T	T	Correct
12	TTT	T	T	N	T	N	T	T	N	N	N	Correct

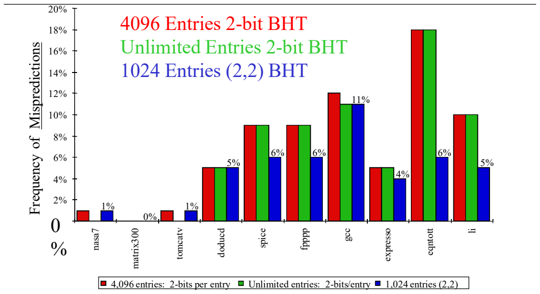
Hardware Costs of 2-level predictions

- (m,n) predictor ➔ m-bits of global history, n-bit predictor
- 2^m*n*Number of prediction entries
- Say you have m-bits of history (m=2)
- n-bits of predictor per entries (n=2)

(2,2) predictor with 1K prediction entries

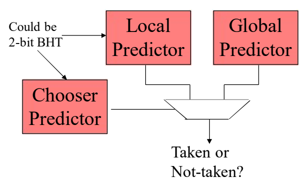
$2^2*2*1024 = 8K$-bits

Accuracy of Different Schemes



Hybrid Branch Predictors

- Tournament predictors: Adaptively combine local and global predictors
- Different schemes work better for different branches



Tagged Hybrid Predictors

- Need methods that combine multiple predictors and then evaluate if the prediction is likely to be associated with the branch.
- This association can depend on branch histories of varying length.
- *According to the textbook it is believed that Intel and others are employing this predictor style in their microprocessors. Though it has not been disclosed publically.*

Tagged Hybrid Predictors

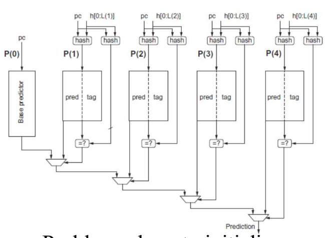
- Need to have predictor for each branch and history
  - Problem: this implies huge tables
  - Solution:
    - Use hash tables, whose hash value is based on branch address and branch history
    - Longer histories may lead to increased chance of hash collision, so use multiple tables with increasingly shorter histories

64

Tagged Hybrid Predictors

Branch Prediction

- Example 5 component predictor – 5 predictor tables
- Accessed with a hash of the PC and history of last *i* branches (like in gshare)
- Each table has different history lengths
- Use of a short tag to determine a match
- Use field indicates if the prediction was recently used and helps with match
- Typically 32-64KiB in size

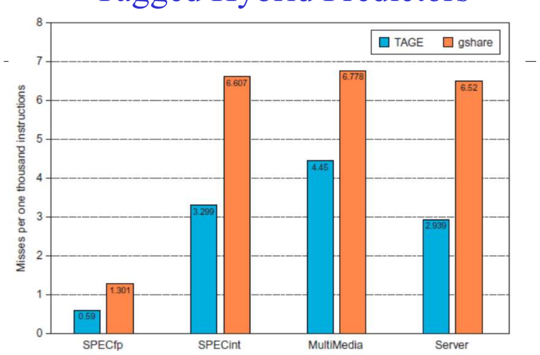


- Problems: how to initialize, need to regularly reset history
- Could use hints from software

65

Tagged Hybrid Predictors

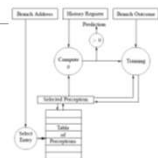
Branch Prediction

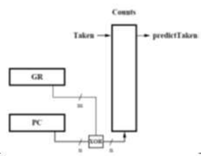


66

Other Branch Predictors

- There are many branch predictors developed over the years.
- We will read:
  - Perceptron Branch Predictor
- We will implement
  - gshare





Branch prediction costs

- The BTB and BHT are big arrays that run every single cycle.
  - They are on critical paths, and thus must run fast.
  - They can be big power hogs.
  - They will never work well if there are data-dependent branches.
  - For long pipes at high frequencies, a good branch prediction is very necessary.
  - For shorter pipes (where the mis-predict penalty is less severe), you can get away with a less elaborate predictor (e.g., smaller arrays).

More costs

- “When you come to a fork in the road, take it”
  - Yogi Berra
- The central idea of branch prediction is that doing nothing is bad.
  - Don’t stall. Instead, s*peculatively execute* a (hopefully pretty good) prediction
  - But executing instructions takes energy, and if you have to flush them then the energy was wasted.
- Unless you’re predictors are *always* right, then branch prediction + speculative execution *always* wastes energy in flushed instructions.
- Nonetheless, the tradeoff is *usually* a good one.