ECSE 425 Lecture 12: Branch Prediction

H&P Chapter 2

Administrative Notes

- Homework
  - Homework 1 back today (grades also on WebCT)
  - Homework 2 due Monday
  - Homework 3 out Monday, due October 17
- Midterm 1
  - 50 minutes, in class, October 12
  - Chapter 1, Appendix A, Chapter 2.1-2.3

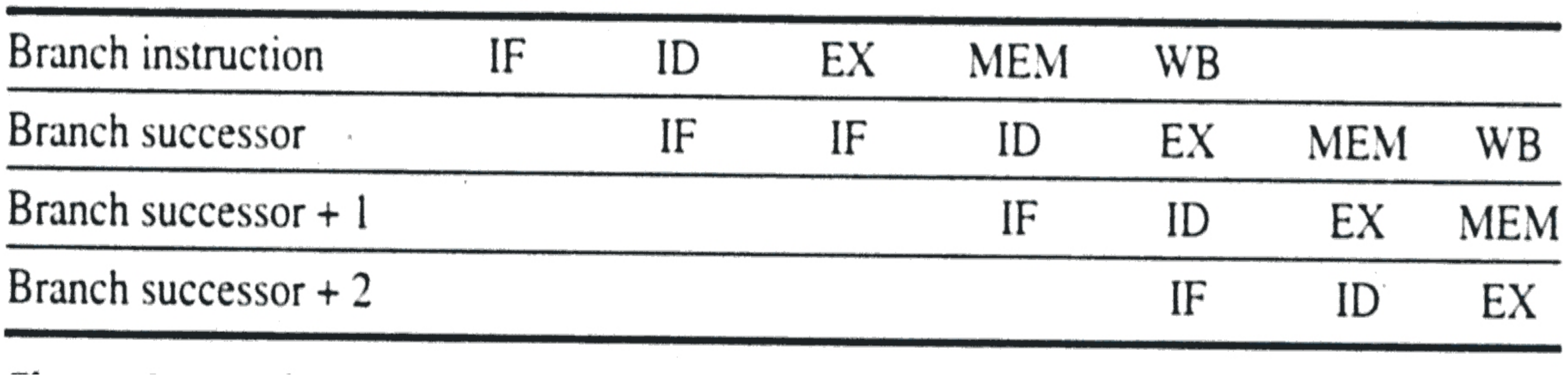
Last Time

- Loop Unrolling

Today

- Chapter 2.3: Branch Prediction
- Static prediction
- Dynamic prediction
  - – 1-bit branch predictors

Why Predict Branches?



Branch instruction	IF	ID	EX	MEM	WB		
Branch successor		IF	IF	ID	EX	MEM	WB
Branch successor + 1				IF	ID	EX	MEM
Branch successor + 2					IF	ID	EX

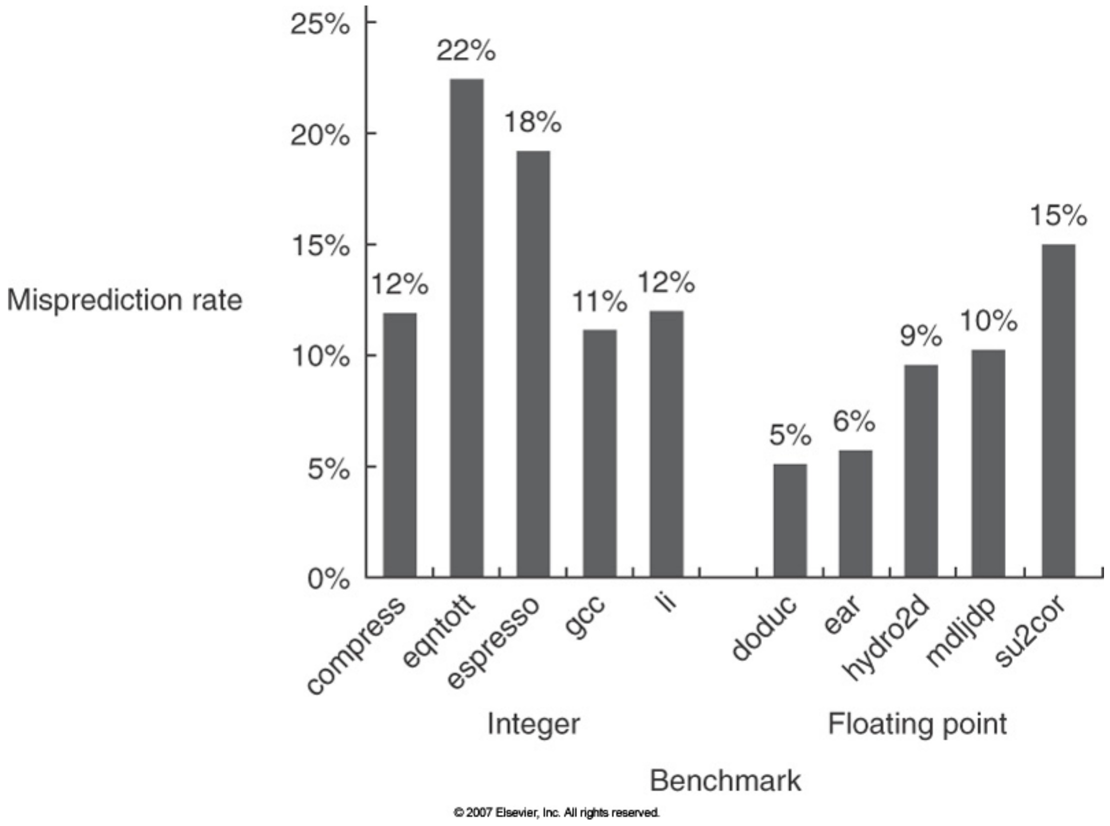
- Problem: branch hazards
  - At least one cycle delay to determine branch direction
- Solution: Make an informed guess
- Guess right? Reduce (or eliminate) the delay!
- Guess wrong? Penalty is no worse than before

Static Branch Prediction Strategies

- Predict-taken
  - Misprediction rate = untaken branch frequency
  - SPEC: 34% misprediction on average (9% to 59%)
- Predict based on branch direction
  - Predict forward-going branches (if-else) as not taken
  - Predict backwards-going branches (loops) as taken

Using Profiling to Assist Prediction

- Compile, profile, re-compile
  - Gather frequent code path data
  - Use it to organize code, provide hints
- Still inaccurate
  - SpecINT: ≤ 22%
  - SpecFP: ≤ 15%

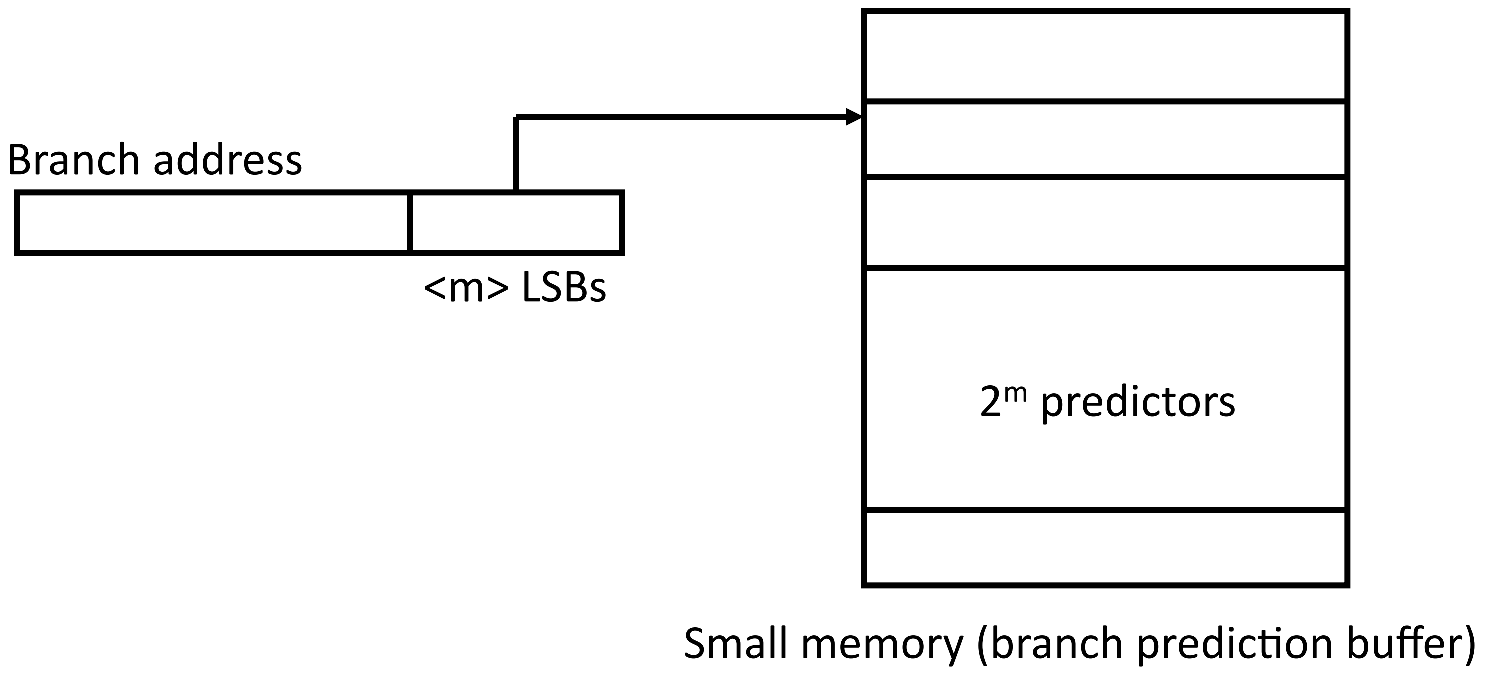


Why Dynamic Branch Prediction?

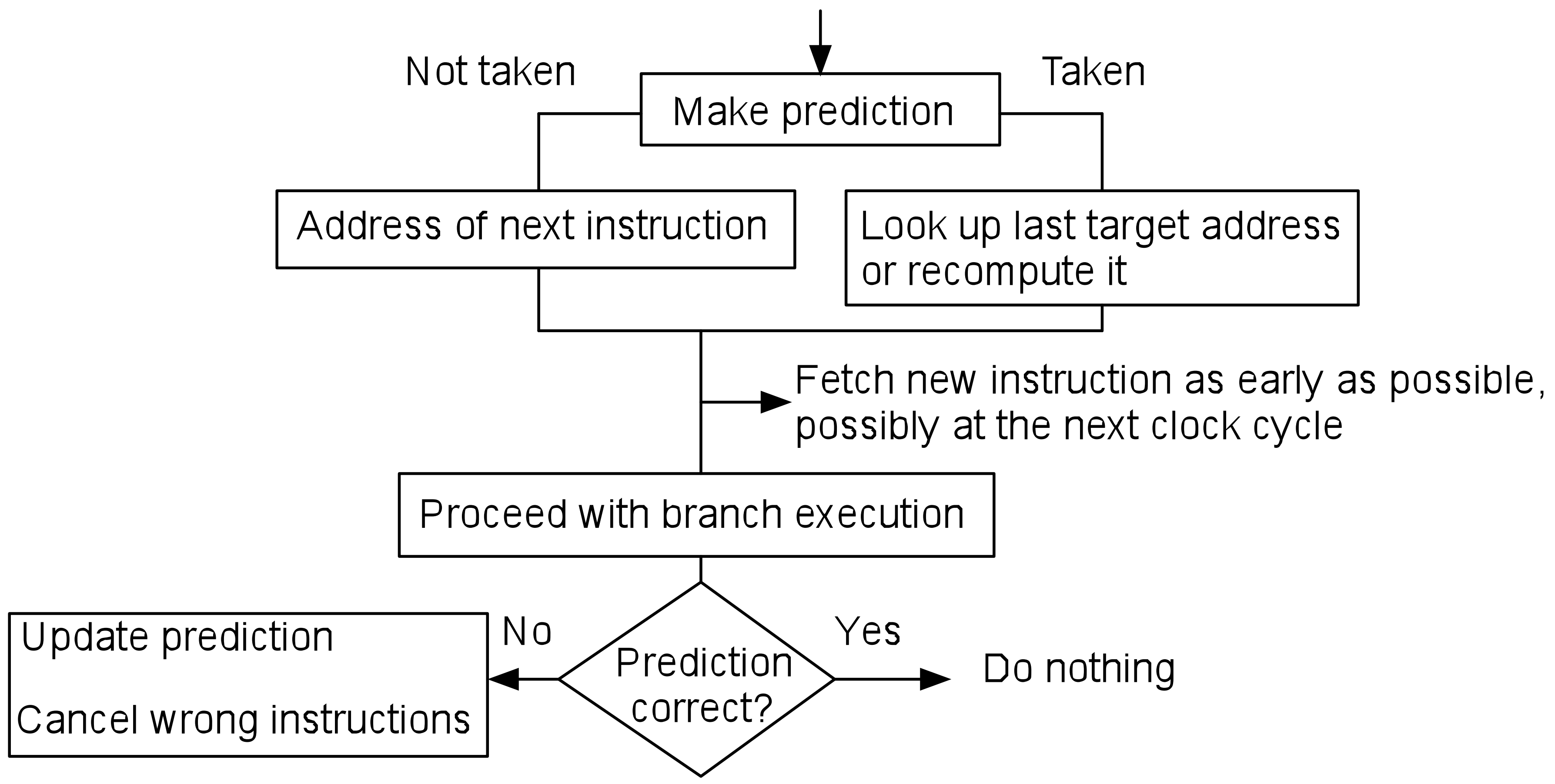
- Branch delays are an obstacle to performance
  - Branches represent ~20% of instructions
  - Deeper pipelines ⇒ longer branch delays!
- Branch prediction performance depends on branch prediction accuracy
  - Static prediction is inaccurate, even with profiling
- Technology trends
  - When life gives you transistors,
  - Make history-based branch predictors!

Dynamic Branch Prediction

- *Idea*: predict the outcome of a branch based on its past behaviour



Dynamic Branch Prediction


Not taken
Make prediction
Taken
Address of next instruction
Look up last target address or recompute it
Fetch new instruction as early as possible, possibly at the next clock cycle
Proceed with branch execution
Update prediction
Cancel wrong instructions
No
Prediction correct?
Yes
Do nothing

Performance = *f*(accuracy, cost of misprediction)

Four Branch Prediction Schemes

1. 1-bit Branch Predictor
2. 2-bit Branch Predictor
3. Correlating Branch Predictor
4. Tournament Branch Predictor

1-bit Predictors

- Branch History Table
  - Indexed by LSBs of PC address
  - Table contains 1-bit values: branch last taken or not?
  - No address check (saves HW; causes aliasing)
  - Works for numerical code with many loops
- In a loop, 1-bit BHT is likely wrong twice
  - Last iteration: predicts the loop continues
  - First iteration: predicts exit instead of looping

1-bit Predictor Example

- Loop with 10 iterations

```
Iteration #:                1  2  3  4  5  6  7  8  9  10
Actual branch outcome:      T  T  T  T  T  T  T  T  T  N
Predicted branch outcome:   N  T  T  T  T  T  T  T  T  T
```

- Mispredict twice for every 10 iterations
  - (Assuming that when the branch first executed we predict that it is not taken)
  - 80% prediction accuracy

Updating 1-bit predictors

Iteration	Predictor Bit	Predicted Outcome	Actual Outcome	Update
1	N	N	**T**	**T**
2	T	T	T	T
3	T	T	T	T
4	T	T	T	T
5	T	T	T	T
6	T	T	T	T
7	T	T	T	T
8	T	T	T	T
9	T	T	T	T
10	T	T	**N**	**N**

1-bit predictors

- Problem: 1-bit prediction is wrong whenever there is a transition in the branching pattern
- Example: NTNTNT
  - 1-bit predictor is never correct! (0%)
  - Tossing a coin (no prediction at all) gives 50%!
- However, real code has bias
  - Branches taken several times are likely taken again
- Solution: store more history
  - Try two bits!

Summary

- Static prediction
  - Deterministic, given an application
  - Inaccuracy leads to performance penalties
- Dynamic prediction
  - Attempts to learn the branching pattern
  - 1-bit predictors: may be wrong twice per loop!

Next Time

- More Branch Prediction
  - Chapter 2.3
- On Wednesday, Dynamic Scheduling
  - Chapter 2.4