Detailed Project Report: Custom TAGE Branch Predictor Implementation

Name: Abhinandan Sharma
PID: A69041851

Introduction

Branch prediction is a critical technique used in modern processors to improve instruction-level parallelism by guessing the outcome of conditional branches. This project implements several branch predictors for evaluation, including a custom predictor based on the TAGE (Tagged Geometric History Length) algorithm, one of the most accurate dynamic branch predictors widely recognized in computer architecture research.

Custom Predictor Choice: TAGE

The TAGE predictor is chosen for its advanced predictive capabilities over classic predictors such as Gshare or Tournament. It is designed to efficiently exploit different lengths of global branch history, providing robustness across diverse branch patterns. The key insight behind TAGE is that different branches benefit from different history lengths — a simple heuristic branch predictor cannot adapt to all cases.

- **Geometric History Lengths**: Multiple predictor tables each indexed with histories of geometrically increasing length capture short- and long-range branch correlations.
- **Tagging**: Each table entry contains tags derived from PC and history hashes to reduce aliasing, distinguishing entries that would otherwise collide.
- **Usefulness Counters**: Adaptively disable harmful entries by tracking their usefulness, allowing the predictor to replace stale or less useful entries dynamically.

Implementation Details

Data Structures and Parameters

- **Total Tables:** 5 predictor tables, labelled T0 (base), T1, T2, T3, and T4.
    - T0: Base predictor with no tagging, indexed solely by PC bits.
    - T1–T4: Tagged tables indexed by PC and folded global history bits.
- **History Lengths for Tables:**
    - T0 uses 11 PC bits.
    - T1 to T4 use global histories of lengths 8, 16, 32, and 64 bits respectively.
- **Tag Sizes:**
    - T1 and T2 use 9-bit tags.
    - T3 and T4 use 9-bit tags.
- **Counters:**
    - 3-bit saturating counters for prediction states (values range from strongly not taken to strongly taken).
    - 2-bit usefulness counters per entry for managing allocation and aging.
- **Global History Register:** 64-bit register maintaining recent branch outcomes.

- **Reset Period:** Usefulness counters are periodically reset every 2,097,152 (2048K) predictions to prevent saturation and preserve adaptability.

Indexing and Tag Computation

- Indices for each table are calculated by XOR folding the PC and folded global, minimizing aliasing.
- Tags are computed with a mix of PC bits and branch history bits, folded with bitwise XOR operations with strides and offsets to diversify their patterns.

Prediction Logic

- On prediction, each tagged table is queried from the longest to shortest history (T4 to T1).
- If a tag-match is found in any tagged table, the corresponding predictor's state is used.
- If no tagged entry matches, the base table (T0) provides the prediction.
- The provider component is the tagged table with the longest history length that matches the current PC and history tag.
- If the provider entry is "fresh" (determined by a usefulness counter being zero and the prediction counter in a weak state), the predictor defers to the alternate prediction.
- The alternate prediction is the prediction from the next best matching table with a shorter history length or the base table if no other matches exist.
- This scheme improves accuracy because stale or untrained entries in longer history tables can sometimes be misleading before they are fully trained.
- The alternate predictor thus provides a more confident "backup" decision until the provider entry is deemed reliable.

Training Logic

- The predictor updates the saturating counters for the provider table based on the actual outcome (branch taken or not).
- If the provider's prediction differs from an alternate prediction (next-longest matching table), the usefulness counter is updated to reflect accuracy.
- On a misprediction, the predictor attempts to allocate new entries in tables with longer histories and zero usefulness to capture new patterns.
- If no entries are eligible for allocation, usefulness counters of longer-history tables are decremented as a form of aging to allow subsequent allocation.
- The global history register is updated by shifting in the latest branch outcome.

Usefulness Reset Mechanism

- Halfway through the reset period, the most significant bit (MSB) of usefulness counters is cleared, preserving some information but softening confidence.
- At the full reset period, the least significant bit (LSB) is cleared, resetting usefulness counters more aggressively to avoid saturation and allow adaptation to new branch behavior.

Design Intuition and Benefits

- **Capturing Diverse Behavior**: The geometric increase in history lengths allows capturing both short-range and long-range correlations in branch patterns.
- **Reducing Aliasing:** Tagging with a combination of PC and history bits reduces prediction interference from unrelated branches hashing to the same table entry.

- **Dynamic Learning:** Usefulness counters guide the replacement policy, efficiently aging out entries that do not contribute to prediction accuracy.
- **Adaptivity:** Periodic resetting of usefulness counters ensures the predictor can respond to dynamic program phase changes.
- **Safety Net via Base Predictor:** The base predictor acts as a fallback for cases where no more extended history tables can provide confident predictions.

TAGE Architecture

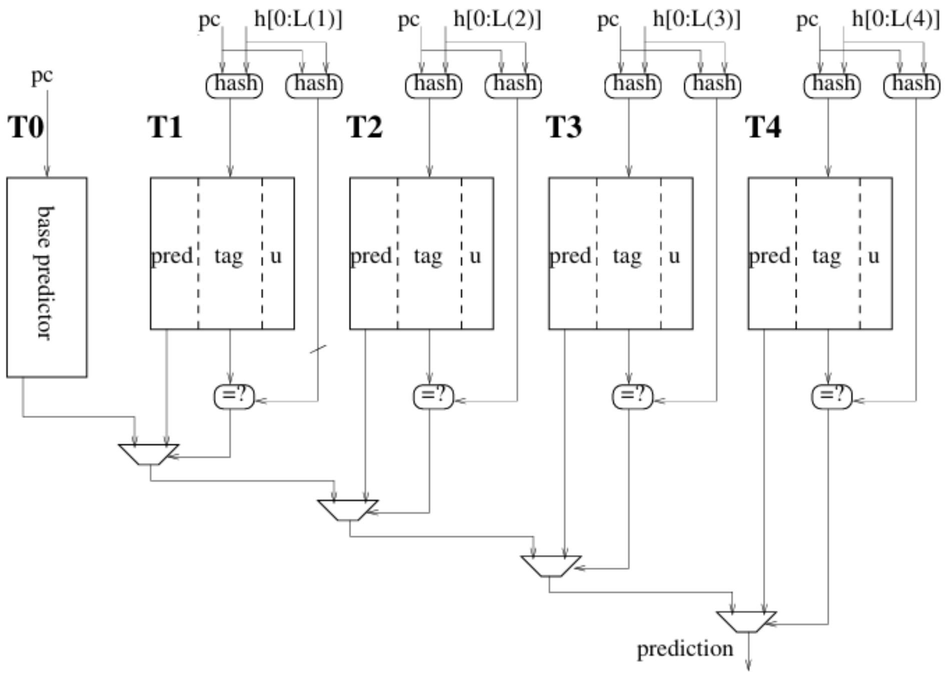

Figure 1. A 5-component TAGE predictor logical synopsis: a base predictor is backed with several tagged predictor components indexed with increasing history lengths. On an effective implementation, predictor selection would be performed thrrough a tree of multiplexors

---

Comparison between different Branch predictors:

S.No	Trace	Gshare	Tournament	Custom (TAGE)
1	U1_Blender.bz2	33.654	29.396	**21.108**
2	U2_Leela.bz2	101.729	96.506	**77.781**
3	U3_GCC.bz2	19.608	15.415	**5.674**
4	U4_Cam4.bz2	10.030	6.416	**6.075**

Budget Allocation

Component	Table Length (entries)	Bits per Entry	Total Bits Used	Description
Base Table (T0)	211=20482^{11} = 2048211=2048	3 (3-bit saturating predictor counter)	6144	Simple predictor counters per entry
Tagged Table (T1)	210=10242^{10} = 1024210=1024	3 (predictor) + 9 (tag bits) + 2 (usefulness) = 14	14,336	Tagged predictor entry with usefulness counter
Tagged Table (T2)	210=10242^{10} = 1024210=1024	3 + 9 + 2 = 14	14,336	Same as T1
Tagged Table (T3)	210=10242^{10} = 1024210=1024	3 + 9 + 2 = 14	14,336	Same as T1
Tagged Table (T4)	210=10242^{10} = 1024210=1024	3 + 9 + 2 = 14	14,336	Same as T1
Global History Register	1	64	64	64-bit global history tracking
Reset Counter	1	21 (to count up to 2,097,152)	21	Tracks training iterations for utility reset
Total			61,529 bits (~7.56 KB)	Aggregated storage for predictor tables and associated data

References

1. André Seznec. "**Analysis of the O-GEometric History Length branch predictor.**"
   IRISA/INRIA/HIPEAC, Campus de Beaulieu, 35042 Rennes Cedex, France.
2. "**The Alpha 21264 Microprocessor.**"
   Description of high-performance features and speculative execution in Alpha 21264.
3. André Seznec, Pierre Michaud. "**A case for (partially) TAgged GEometric history length branch prediction.**"
   IRISA/INRIA/HIPEAC, Campus de Beaulieu, 35042 Rennes Cedex, France.