Vol.2011-ARC-196 No.21
2011/7/28

多様な履歴の利用による分岐予測精度の向上

畔柳圭佑[†1]　石井康雄[†1,†2]
稲葉真理[†1]　平木敬[†1]

現代の高性能なプロセッサには高精度な分岐予測器の存在が欠かせない．近年では複数の予測器を組み合わせ，長い履歴を活用できる予測器が多数が提案されている．一方で，どのような履歴を利用するのが予測精度向上に貢献するのかという研究は最近ではほとんど行われていない．本稿では Set 分け方法によって多様な性質を持つ Per-Set 履歴に着目し，その Set 分けの方法と組み合わせに関して提案し評価を行った．複数の Set 分け方法で生成された Per-Set 履歴を組み合わせて利用することで．単一の方法のみを利用する場合と比較して予測ミス率を 1.0%削減出来た．

Improving Branch Prediction Accuracy by Using Various Histories

Keisuke Kuroyanagi,[†1] Yasuo Ishii,[†1,†2]
Mary Inaba[†1] and Kei Hiraki [†1]

High performance processors require accurate branch predictors. Recently, many predictors that combine multiple predictors and utilize very long history have been proposed. However, there are few studies about what histories should we use. We focused on the Per-Set history that can have various characters depending on the classify method and proposed classify methods for Per-Set history. Using Per-Set histories classified by multiple methods is 1.0% more accurate than using Per-Set histories classified by the single method.

†1 東京大学情報理工学系研究科
Guraduate School of Information Science and Technology, The University of Tokyo
†2 日本電気株式会社
NEC Corporation

1. はじめに

広い実行幅と深いパイプラインを持つ高性能なプロセッサの構成には高精度な分岐予測器の存在が欠かせない．そのため，効率的で高精度な予測を目指してこれまで多くの分岐予測器が提案されてきた．その多くはすべての分岐結果を並べたものである Global 履歴，命令ごとの分岐結果を並べた Local 履歴，これまで実行してきた経路の情報である Path 履歴のいずれか，もしくはその組み合わせを利用している．

近年提案されている高精度な分岐予測器は Perceptron[1)]，O-GEHL[2)]，TAGE[3)] のようにそれぞれ異なった履歴を利用する多数のテーブルを組み合わせ，長い履歴から効率よく関連性を引き出す機構を備えている．例えば 2004 年に開催された Championship Branch Prediction[4)] にて Best Practice Award を受賞した O-GEHL[2)] は，8 個のテーブルを組み合わせ，長さ 200 の Global 履歴を利用している．それぞれのテーブルは異なる長さの Global 履歴を用いて Two-Level Adaptive 予測器[5)] や Gshare 予測器[6)] と類似した方法で利用される．テーブル単体に注目すれば，履歴長 200 の Gshare 予測器の精度は履歴長 10 数程度の Gshare 予測器のそれに大きく劣るが，O-GEHL は利用する履歴をうまく組み合わせることで高い予測精度を達成している．つまり，履歴を利用する機構が進歩したため，有効に利用できる履歴の種類も変化したと言うことができる．

近年の予測器で有効利用できる可能性のある履歴に Per-Set 履歴[7)] がある．Per-Set 履歴は分岐命令を何らかの方法によって複数の Set に分け，それぞれの Set ごとに記録される履歴である．Per-Set 履歴を利用することが精度向上につながるケースが存在することが指摘されており[8)]，実際に FTL++[9)] は Per-Set 履歴を利用し高い精度の予測を実現している．しかし，FTL++は命令アドレスの下位ビットを利用した単純な Set 分け方法のみを採用しており，多様な履歴を利用するという観点からは構成の探索が不十分であり，Per-Set 履歴持つ利点を十分に引き出せていない．

本稿では Per-Set 履歴の Set 分け方法に焦点を当てる．どのような性質を持った履歴の利用・組み合わせが高精度な予測に役立つのか調査を行うため，Per-Set 履歴の Set 分け手法を提案し，評価を行う．まず，2 章において関連研究と Per-Set 履歴の利用が有効になってきた背景について述べる．3 章で Per-Set 履歴の Set 分け方法の候補を提案し，4 章で評価・考察を行う．5 章がまとめとなっている．

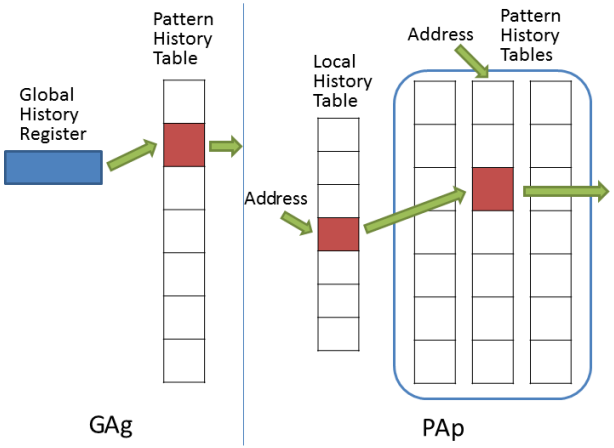

図 1 Two-Level Adaptinve 分岐予測器 (GAg,PAp)
Fig. 1 Two-Level Adaptive branch predictor (GAg,PAp)

2. 関 連 研 究

2.1 Two-Level Adaptive 分岐予測器

プログロラム中に存在する分岐の多くは過去の分岐結果と相関を持つため，分岐履歴を利用することで効率的に分岐予測を行うことができる．履歴を利用する分岐予測器として，Yeh らは Two-Level Adaptive 分岐予測器[5] を提案した．Two-Level Adaptive 分岐予測器は**図 1** のように分岐履歴を利用して Pattern History Table にアクセスし，そこに格納されている 2bit カウンタに基づいて予測を与える予測器である．Two-Level Adaptive 分岐予測器に関する研究[7] ではすべての分岐結果を並べた Global 履歴，一般に利用されている Local 履歴の元になっている分岐命令アドレスごとの分岐結果である Per-Address 履歴，分岐命令を何らかの方法で複数の Set に分けそれぞれの Set ごとに記録される Per-Set 履歴の利用が提案されている．Per-Set 履歴の Set 分け方法としては，命令アドレスの一部による Set 分けの他にも Op コードやコンパイラの付与した情報による Set 分けが提案されている[7] が，命令アドレスの下位 bit によるもののみが評価対象となっている．

2.2 Perceptron 分岐予測器

Perceptron 分岐予測器[1] は Jiménez らによって提案された予測器である．**図 2** のように

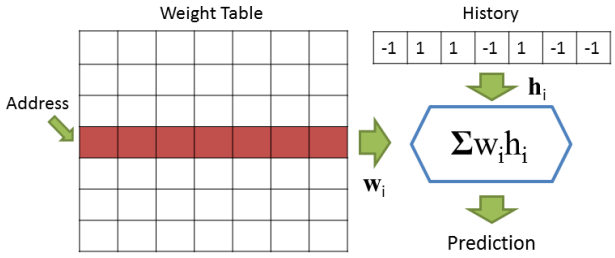

図 2 Perceptron 分岐予測器
Fig. 2 Perceptron branch predictor

利用する履歴 1bit ずつに対して重み付けをし，その重みを足し合わせて合計値の符号を予測とする．利用する履歴長に対して必要な記憶容量が線形となるという特徴がある．それに対し，Pattern History Table を用いる Two-Level Adaptive 分岐予測器などの予測器では指数オーダーで必要な容量が増加する．このため，Perceptron 予測器は数 10 程度と以前の予測器よりも長い履歴を利用した予測を行うことができる．また，提案時に指摘された通り Global 履歴以外の履歴も自然に組み合わせることができる．

2.3 Geometric History Length

Geometric History Length[2] を利用する予測器は各テーブルで利用する履歴長が等比数列になるような構成を取る．Perceptron 分岐予測器の履歴 1bit に対してテーブル 1 つが必要であり，予測のたびに多くのカウンタを足し合わせる必要があるという欠点をこれにより改善している．また，必要なハードウエア容量は履歴長に対して対数オーダーとなり，Perceptron 分岐予測器よりもさらに長い履歴を利用して予測を行うことができる．この手法を採用する代表的な予測器に O-GEHL と TAGE が存在する．

2.3.1 O-GEHL

O-GEHL[2] は Seznec により提案された，Geometric History Length を初めて利用した予測器である (**図 3**)．命令アドレスと各テーブルに対して等比数列になるように設定された履歴長の履歴をハッシュし，テーブルのインデックスを生成する．これを使って Two-Level Adaptive 分岐予測器と同じようにテーブルにアクセスする．テーブルにはカウンタが格納されており，Perceptron 分岐予測器と同様にこれらの値を足し合わせ合計値の符号を予測として利用する．O-GEHL は長さ数百程度の履歴を利用して予測を行うことができる．この手法をベースとした予測器に，後述する Fused Two-Level Adaptive Branch Predictor[10] がある．

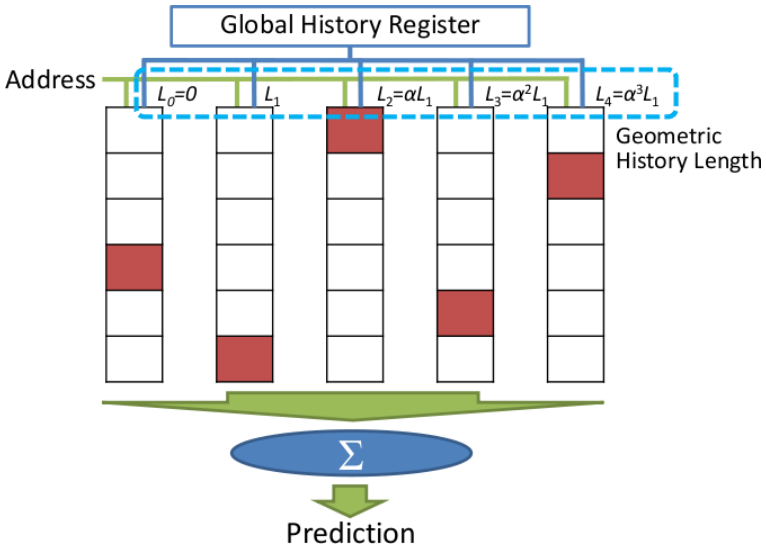

図 3 O-GEHL 分岐予測器
Fig. 3 O-GEHL branch predictor

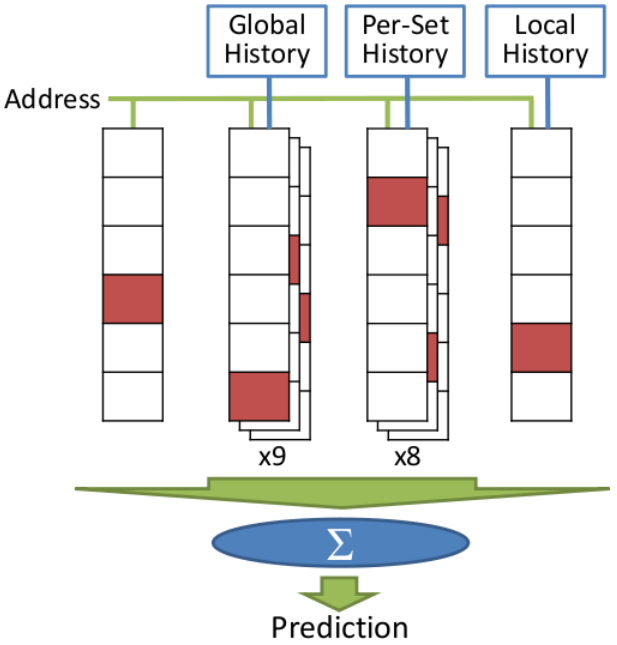

図 4 FTL++分岐予測器
Fig. 4 FTL++ branch predictor

2.3.2 TAGE

TAGE[3] は Seznec らによって提案された予測器であり，各テーブルに対して命令アドレスと履歴からハッシュ値を 2 つ生成し，一つをテーブルのインデックスに，もう一つをテーブルが保持しているタグとの比較に利用する．タグのマッチしたテーブルの中で最長の履歴を利用するものを予測として採用する．これにより異なった命令・履歴の組み合わせがテーブルの同じエントリを参照してしまうエイリアシングを大幅に削減することができ，より長い履歴を有効に利用できる．実際に TAGE をベースとした ISL-TAGE[11] は 2000 近い履歴長を利用している．

2.4 Fused Two-Level Adaptive Branch Predictor

Fused Two-Level Adaptive Branch Predictor[10] (FTL) は石井によって提案された，O-GEHL をベースとしている予測器で 2006 年に開催された 2nd Championship Branch Prediction[12] で第 2 位，FTL を改良した予測器である FTL++が 2011 年の 3rd Championship Branch Prediction[13](CBP3) で第 2 位と高精度な予測を実現している． FTL++は**図 4** のように複数の種類の履歴を組み合わせて予測を行う．それぞれ異なった種類や長さの履歴を使いテーブルにアクセスし，格納されているカウンタの値を足し合わせ，O-GEHL と同様に合計値の符号を予測として利用する．FTL++は Per-Set 履歴を利用しているが，アドレスの下位ビットを利用した単純な Set 分けのみを採用しており，Set 分け方法を工夫することでさらなる高精度な予測を実現できる可能性がある．

3. Per-Set 履歴

Per-Set 履歴の利用は Two-Level Adaptive 分岐予測器のような単純な予測器においては Global 履歴や Local 履歴の利用と比較して精度の点で劣っていた[7]．そのため，Per-Set 履歴は提案時以降ほとんど注目されてこなかった．しかし，前章で述べたように近年では複数の履歴を組み合わせて利用できる機構を備えた予測器が多数提案されており，そのような分岐予測器においては Set 分けの方法によって多様な履歴を生成できる Per-Set 履歴の利用

が有効である可能性がある．

3.1 Set 分類の方法

本稿では有効な Per-Set 履歴の種類を探索するため，以下のとおり 4 種類の方針に基づいて 8 種類の Set 分け方法を考案した．以下では説明のため，履歴テーブルのエントリ数を N，インデックスに利用する bit 数を L とする ($N = 2^L$)．

方針 1　命令アドレスを利用する

アドレス下位ビット (A)　命令アドレスの下位 Lbit を利用する．Yeh らの研究や FTL++ではこの方法を利用して Set 分けを行なっている．

アドレス下位ビット (AA)　命令アドレスの下位 $2L$bit を利用する．最下位 Lbit と次の Lbit との排他的論理和を取りインデックスとする．

これらは命令アドレスの下位ビットを利用する一般的な手法である．二つの手法でエントリの衝突の仕方が異なるため，組み合わせて利用した際に有効である可能性が高い．

方針 2　命令の性質を利用する

分岐方向 (D)　命令アドレスと分岐先アドレスを比較し，前方への分岐か後方への分岐か判別する．分岐方向の情報 1bit と命令アドレスの下位 $L-1$bit をつなげてインデックスとする．

分岐先上位ビット一致 (S)　命令アドレスと分岐先アドレスの上位ビットを比較する．本稿では命令アドレス 32bit 中の上位 20bit を比較する．一致したか否かの情報 1bit と命令アドレスの下位 $L-1$bit をつなげてインデックスとする．

これらは命令アドレスの下位ビットに加えて，1bit 別の情報を使用している．前方への分岐はループを制御する分岐であることが多く，静的分岐予測でも利用されるように後方へ分岐する命令と性質が異なる場合が多い[14]．また，命令アドレスと分岐先アドレスの上位 bit が異なる分岐は分岐先が離れているものであることが多く，そのような分岐は実行する命令の流れを大きく変える命令である可能性が高い．そのような分岐命令を他の分岐命令とは別のエントリに割り当てることが有効であるか，この手法を利用することで確認できる．

方針 3　履歴を利用する

Global 履歴 (G)　Global 履歴 Lbit をインデックスとする．

Glabal 履歴+アドレス下位ビット (GA)　Global 履歴 Lbit と命令アドレス下位 Lbit の排他的論理和をインデックスとする．

Path 履歴 (P)　Path 履歴 Lbit をインデックスとする．Path 履歴として実行てきた分岐命令のアドレス最下位ビットを並べたものを利用する．

これらは他の手法と異なり，一つの命令が状況によって異なる Set に割り当てられる．履歴によって大きく性質が変わる命令に対してはこのような手法が有効である可能性がある．

方針 4　分岐先情報を利用する

ターゲット (T)　分岐先アドレスの下位 Lbit を利用する．

各命令にもともと含まれている情報を利用するという点で命令アドレスを利用するのもと類似した方針であるが，ループからの脱出のための分岐命令が同じエントリを参照するなど，命令アドレスによる手法と比べエントリが衝突する理由がプログラムの構造に基づいていることが多いと考えられる．

4. 評価と考察

4.1 評 価 環 境

Per-Set 履歴を用いる代表的な高精度分岐予測器である FTL++を用いて Per-Set 履歴の Set 分け方法，その組み合わせについて評価を行う．FTL++の Per-Set 履歴・Local 履歴を利用するテーブルを，すべてエントリ数 32 の Per-Set 履歴を利用するテーブルに置き換えた予測器を利用する．評価に用いた予測器の詳しい構成を**表 1** に示す．

表 1　評価に用いた分岐予測器の構成
Table 1　The configuration of the branch predictor that was used for evaluation.

容量	65KB
BIM	1 table,4K-entry/table,6-bit counter
Global	9 tables, 履歴長 6-293 (Geometric history length)
	4K-entry/table(8),1K-entry/table(1),6-bit counter
Per-Set	10 tables, 履歴長 4-55 (Geometric history length)
	4K-entry/table,6-bit counter

評価環境としては CBP3 の Simulation Infrastructure を利用する．これは，トレースベースのシミュレータで 40 個のトレースが用意されており，1 トレースあたり 50Muops が実行される．評価基準として全トレースにおける千命令あたりの予測ミス数 (MPKI) の平均値を採用する．

4.2 Set 分け方法の影響

各 Set 分け方法を用いた Per-Set 履歴を利用し，Set 分けの性能への影響を調査する．

表 2 各 Set 分け方法を使用した場合の予測精度 (MPKI 値).
Table 2 Prediction accuracy (MPKI) for each classify methods.

Set 分け方法	MPKI
アドレス下位ビット (A)	4.214
アドレス下位ビット (AA)	4.214
分岐方向 (D)	4.247
分岐先上位ビット一致 (S)	4.256
Global 履歴 (G)	4.432
Glabal 履歴+アドレス (GA)	4.428
Path 履歴 (P)	4.447
ターゲット (T)	4.300

結果を**表 2** に示す．アドレス下位ビットによる 2 種類の手法が同程度で最も精度が良かった．アドレス下位ビットをインデックスに利用することで，長い履歴長を利用していることもあり小さいループを実行している間であれば十分に Local 履歴としての役割を果たしていると考えられる．履歴を利用する 3 種類の手法 (G,GA,P) の予測精度が悪い理由も同様に，履歴によって利用するエントリが変わってしまうことで Local 履歴としての役割をうまく果たせなくなってしまうためであると考えられる．

4.3 複数の Set 分け方法の組み合わせ

複数の Set 分け方法を組み合わせた場合の予測精度を測定する．2 種類の組み合わせ 64 通りと 3 種類の組み合わせ 512 通りの構成に対してそれぞれ評価を行なった．組み合わせ方法は，Per-Set 履歴を利用するテーブルのうち，短い履歴を使用するものから順番に別々の Set 分け方法で生成した履歴を利用することとした．全体として利用する履歴長・テーブル数は前実験と変わらない．2 種類の Set 分け方法を組み合わせた場合の結果を**表 3** [⋆1]に，3 種類の Set 分け方法を組み合わせた場合の結果の上位 5 つの構成を**表 4** に示す．

2 種類の Set 分け方法の組み合わせではアドレス下位ビット (A) とアドレス下位ビット (AA) を組み合わせたものが最も良い予測精度を達成している．MPKI は 4.178 と 1 つの Set 分け方法のみ利用時の最良値 4.213 と比較し 0.8%予測ミスが削減されている．また，他手法との組み合わせを見ても，単一の Set 分け手法のみを利用した場合と比較して多くの場合で予測精度が向上している．3 種類の Set 分け方法の組み合わせでは，アドレス下位ビット (AA)・分岐方向 (D)・アドレス下位ビット (A) の組み合わせが MPKI4.171 と最も高精度で単一手法利用時からの改善は 1.0%である．複数の手法を組み合わせることで精度が向上する理由としては，ある一つの Set 分け方法では分岐結果に相関のある履歴を含んでいない履歴テーブルのエントリを参照してしまう場合にも，別の Set 分け方法を組み合わせることで，相関のある履歴を利用できる可能性が高まることが挙げられる．

⋆1 対角線上の数値と表 2 内の数値の違いはアドレスと履歴をハッシュしてインデックスを生成する際の命令アドレスの扱いの違い (ローテート量の違い) に起因している．

表 3 2 種類の Set 分け方法の組み合わせた場合の予測精度 (MPKI 値)．太字は行内最良値，下線は列内最良値.
Table 3 Prediction accuracy (MPKI) for each combination of two types of classify method. The boldface number is the best value in the row and underlined number is the best value in the column.

	A	AA	D	S	G	GA	P	T
アドレス下位ビット (A)	4.213	**4.178**	4.213	4.22	4.224	4.23	4.222	4.214
アドレス下位ビット (AA)	**4.182**	4.214	4.193	4.199	4.216	4.218	4.213	4.204
分岐方向 (D)	4.213	**4.195**	4.248	4.238	4.237	4.246	4.232	4.232
分岐先上位ビット一致 (S)	4.219	**4.198**	4.238	4.257	4.243	4.252	4.241	4.232
Global 履歴 (G)	**4.227**	4.229	4.234	4.242	4.43	4.388	4.364	4.256
Glabal 履歴+アドレス (GA)	**4.227**	4.229	4.246	4.255	4.39	4.431	4.358	4.255
Path 履歴 (P)	**4.217**	4.219	4.228	4.236	4.362	4.350	4.418	4.267
ターゲット (T)	4.214	**4.213**	4.219	4.224	4.252	4.248	4.272	4.304

表 4 3 種類の Set 分け方法を組み合わせた場合の上位 5 構成の予測精度 (MPKI 値).
Table 4 Prediction accuracy (MPKI) of the best five combinations of three types of clmassify method.

Set 分け方法の組み合わせ	MPKI
AA,D,A	4.171
AA,A,A	4.178
AA,S,A	4.179
A,AA,A	4.179
A,AA,AA	4.180

5. おわりに

本稿では Per-Set 履歴の Set 分け手法とその組み合わせが予測精度に与える影響について分岐予測器 FTL++を利用して調査した．実験の結果，Set 分け手法は命令アドレスをベースとしたものが良く，複数の Set 分け方法で生成された Per-Set 履歴を組み合わせて利用することでより高精度な予測を行うことができることが分かった．

この結果は，Perceptron など複数の履歴を組み合わせて利用できる予測器にも適用でき

ると考えられる．現在最も精度の高いと考えられている TAGE ベースの予測器で Per-Set 履歴を利用することによってさらなる予測精度の向上が達成できる可能性があるが，TAGE には各テーブルの出力のうちいずれを採用するか決定するために優先度を与えなければならないという問題がある．例えば，履歴長 100 の Global 履歴と履歴長 30 の Per-Set 履歴のどちらを利用するテーブル優先すべきかは単純には決定できない．このため，多様な履歴を利用した TAGE を構成するためには，効率の良い優先度付けの手法の開発が不可欠である．

本稿では Per-Set 履歴に着目したが，予測したい分岐の結果と相関があれば利用する情報は分岐履歴や Path 履歴でなくとも構わない．実際に Call 命令や Return 命令実行後の領域では分岐結果が履歴とあまり相関しないため，分岐履歴を Call/Return の命令アドレスで置き換える手法が提案されている[15]．しかし，数十以上の長い履歴を扱える高精度な予測器に対してはこの手法は単純には適用できない．このように，プログラムの構造に着目した，柔軟な履歴のあり方に関して研究を行うことは今後の課題である．

また，今回実験に用いた構成の FTL++など，複雑な予測器をそのまま実際のハードウエアに実装するのは難しいケースが多い．現在広く使用されているような，より複雑性の低い予測器において Per-Set 履歴を有効に利用できるケースが存在するか評価することは，現実のプロセッサの性能向上のためには重要である．

参 考 文 献

1) Jiménez, D.A. and Lin, C.: Dynamic Branch Prediction with Perceptrons, *Proceedings of the 7th International Symposium on High-Performance Computer Architecture*, HPCA '01, Washington, DC, USA, IEEE Computer Society, pp.197– (online), available from ⟨http://portal.acm.org/citation.cfm?id=580550.876441⟩ (2001).
2) Seznec, A.: Analysis of the O-GEometric History Length Branch Predictor, *Proceedings of the 32nd annual international symposium on Computer Architecture*, ISCA '05, Washington, DC, USA, IEEE Computer Society, pp.394–405 (online), DOI:http://dx.doi.org/10.1109/ISCA.2005.13 (2005).
3) Seznec, A. and Michaud, P.: A case for (partially) TAgged GEometric history length branch prediction, *The Journal of Instruction Level Parallelism*, Vol.8 (online), available from ⟨http://www.jilp.org/vol8⟩ (2006).
4) CBP: The 1st JILP Championship Branch Prediction Competition. `http://www.jilp.org/cbp/`.
5) Yeh, T.-Y. and Patt, Y. N.: Two-level adaptive training branch prediction, *Proceedings of the 24th annual international symposium on Microarchitecture*, MICRO 24, New York, NY, USA, ACM, pp. 51–61 (online), DOI:http://doi.acm.org/10.1145/123465.123475 (1991).
6) McFarling, S.: Combining Branch Predictors, *Technical Note TN-36, DigitalEquipment Corporation Western Research Laboratories* (1993).
7) Yeh, T.-Y. and Patt, Y.N.: A comparison of dynamic branch predictors that use two levels of branch history, *Proceedings of the 20th annual international symposium on computer architecture*, ISCA '93, New York, NY, USA, ACM, pp.257–266 (online), DOI:http://doi.acm.org/10.1145/165123.165161 (1993).
8) 畔柳圭佑，澤田武男，石井康雄，稲葉真理，　平木敬：分岐予測器におけるローカル履歴テーブルの最適な構成の探索，先進的計算基盤システムシンポジウム論文集，Vol.2011, pp.211–212 (2011).
9) Ishii, Y., Kuroyanagi, K., Sawada, T., Inaba, M. and Hiraki, K.: Revisiting Local History to Improve the Fused Two-Level Branch Predictor, *2nd JILP Workshop on Computer Architecture Competitions* (2011).
10) Ishii, Y.: Fused Two-Level Branch Prediction with Ahead Calculation, *The Journal of Instruction-Level Parallelism*, Vol.9 (2007).
11) Seznec, A.: A 64 Kbytes ISL-TAGE branch predictor, *2nd JILP Workshop on Computer Architecture Competitions* (2011).
12) CBP2: The 2nd JILP Championship Branch Prediction Competition. `http://cava.cs.utsa.edu/camino/cbp2/`.
13) CBP3: 2nd JILP Workshop on Computer Architecture Competitions. `"http://www.jilp.org/jwac-2/"`.
14) Smith, J.E.: A study of branch prediction strategies, *Proceedings of the 8th annual symposium on Computer Architecture*, ISCA '81, Los Alamitos, CA, USA, IEEE Computer Society Press, pp.135–148 (online), available from ⟨http://portal.acm.org/citation.cfm?id=800052.801871⟩ (1981).
15) Porter, L. and Tullsen, D. M.: Creating artificial global history to improve branch prediction accuracy, *Proceedings of the 23rd international conference on Supercomputing*, ICS '09, New York, NY, USA, ACM, pp. 266–275 (online), DOI:http://doi.acm.org/10.1145/1542275.1542315 (2009).