分岐予測器

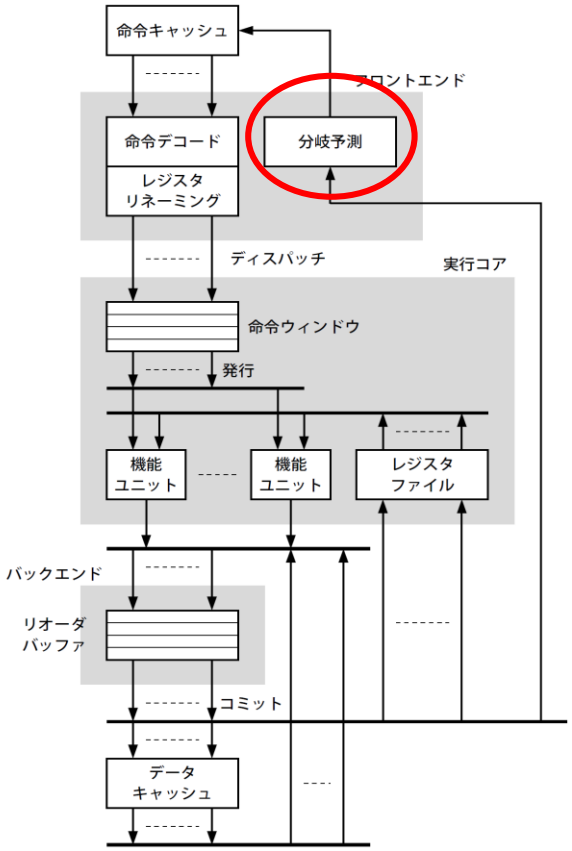
電子情報システム専攻
安藤秀樹

内容

- 目的
- 2ビット・カウンタ方式 [Lee 1984]（復習）
- 2レベル適応型方式
  - PAs [Yeh 1991, Yeh 1992, Yeh 1993]
  - GAs [Pan 1992]
  - gshare [McFarling 1993]
  - 競合の低減手法 [Chang 1996, Sprangle 1997, Lee 1997, Michaud 1997]
- ハイブリッド予測器 [McFarling 1993, Chang 1995, Evers 1996]
- 最新の予測器 TAGE [Seznec 2006]

分岐予測の目的(1)

- 制御ハザードによるパイプライン・ストールを回避

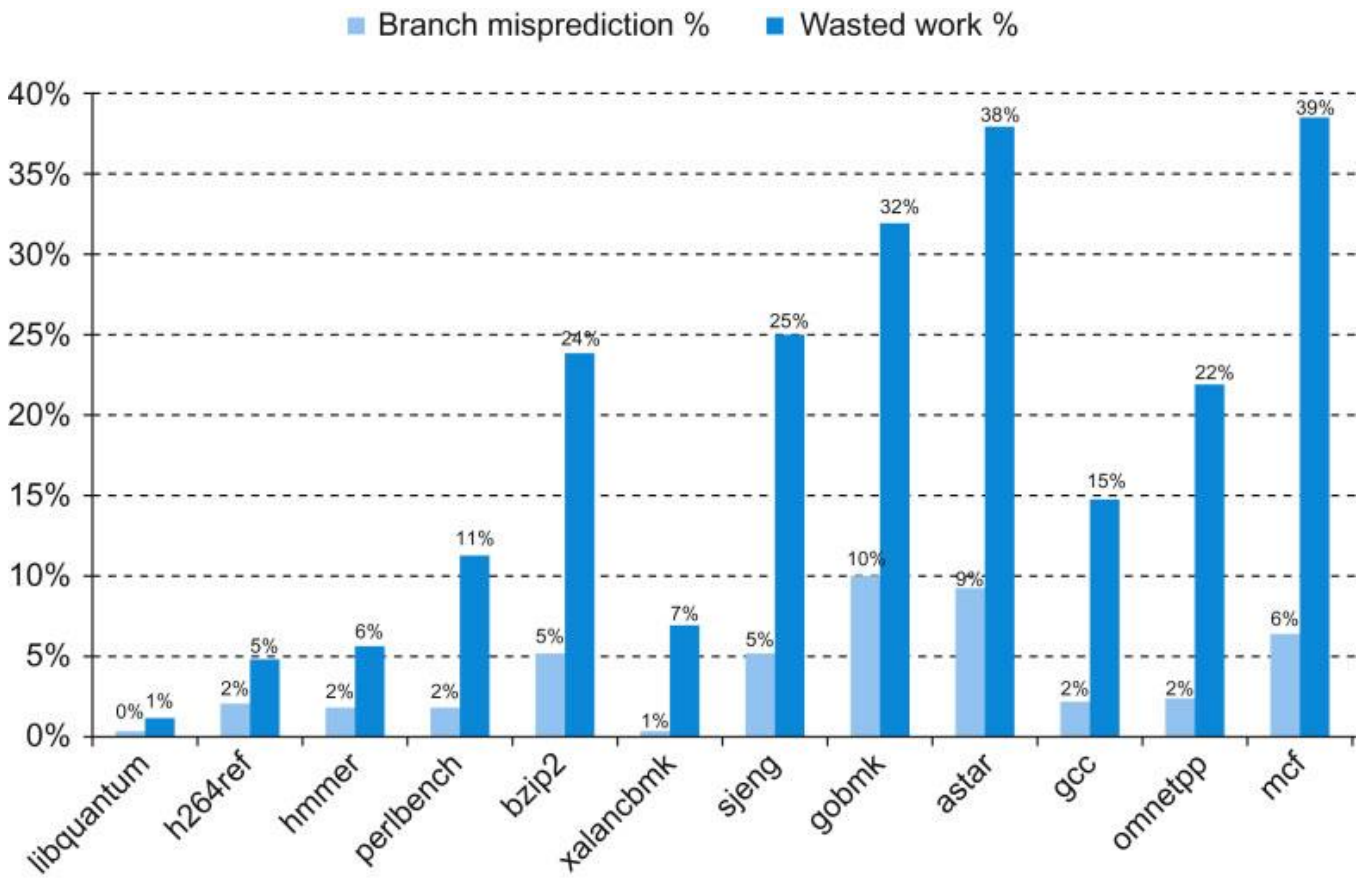


分岐予測の目的(2)

- 深いパイプライン
  - ストール・サイクルは非常に大きい
  - 分岐予測ミス・ペナルティ, 10+サイクル
  - 数%の予測ミスで、数十%の性能低下
  - 非常に多くの無駄な命令の実行
    - 消費電力の増加

- 並列実行
  - サイクル当たりの分岐出現頻度は高い

分岐予測ミス率と
無駄となった命令実行数



D. A. Patterson and John L. Hennessy, Computer Organization and Design, Fifth Edition: The Hardware/Software Interface, Morgan Kaufmann, 2013.
Sandy Bridgeより以前の世代のプロセッサの評価のようである

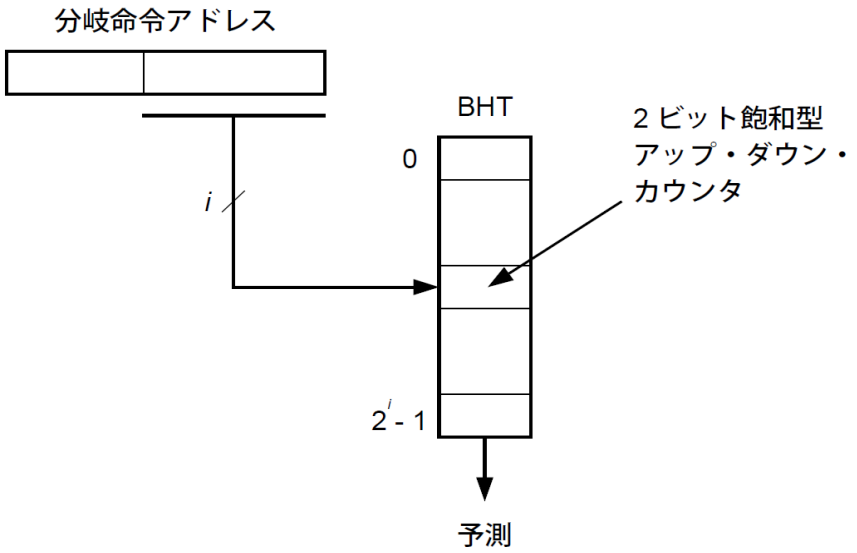
分岐予測の目的(3)

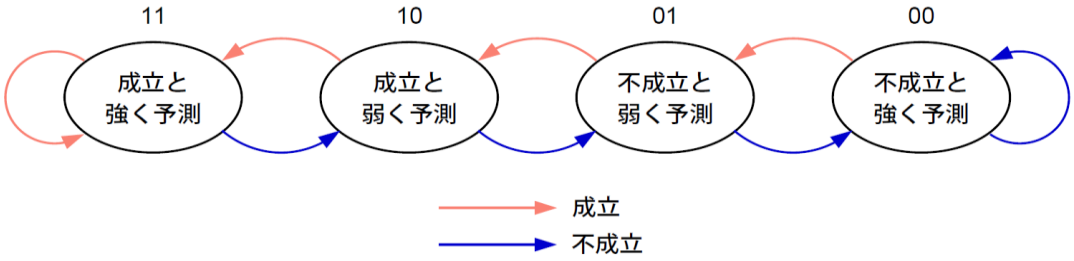
- アーキテクチャの進歩
  - 実行サイクル数が減少するほど分岐予測ミス・ペナルティの性能への影響は大きくなる
- 命令ウィンドウをできるだけ一杯にする
  - 命令が多いほど、並列に実行できる命令の発見確率は高い
  - 特に、メモリ・レベル並列（後の講義）に効く

分岐予測

- 過去の履歴から将来を予測
  - 過去の振る舞いと将来の振る舞いには、強い相関

2ビット・カウンタ分岐予測(2bc) [Lee 1984]





予測精度向上

- 競合を減らす
  - BHT(分岐履歴表:branch history table)のエントリ数増加
    - 良く実行される静的分岐の数をおよそカバーする十分な数
  - 4Kエントリ程度で十分

- 新たな相関
  - 分岐のパターン

内容

- 目的
- 2ビット・カウンタ方式 [Lee 1984]（復習）
- 2レベル適応型方式
  - PAs [Yeh 1991, Yeh 1992, Yeh 1993]
  - GAs [Pan 1992]
  - gshare [McFarling 1993]
  - 競合の低減手法 [Chang 1996, Sprangle 1997, Lee 1997, Michaud 1997]
- ハイブリッド予測器 [McFarling 1993, Chang 1995, Evers 1996]
- 最新の予測器 TAGE [Seznec 2006]

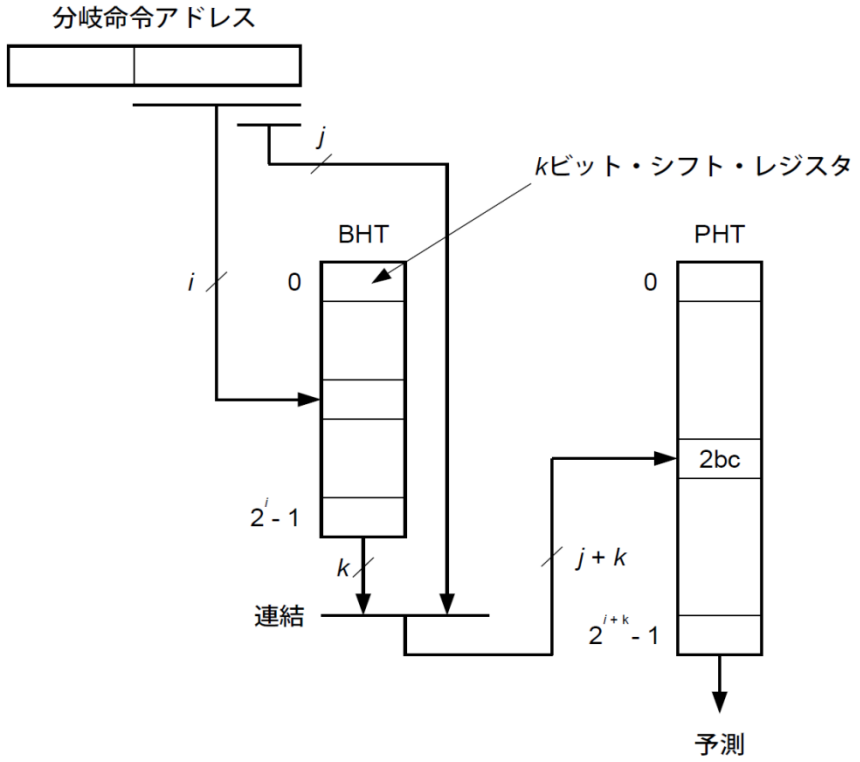
2レベル適応型分岐予測

- 2bc
  - 頻度から予測

- 2レベル予測
  - [Yeh 1991, Yeh 1992, Pan 1992, Yeh 1993]
  - パターンをとらえて予測
  - 例：4回繰り返すループの分岐
    - TTN→T ... ループに初めて入ったとき。1度目の繰り返し
    - TNT→T ... 2度目の繰り返し
    - NTT→T ... 3度目の繰り返し
    - TTT→N ... 4度目の繰り返し。ループを抜ける
  - 各分岐につき、4つのパターンごとに、次の分岐結果の頻度を2ビット・カウンタで記録

ローカル履歴2レベル適応型分岐予測

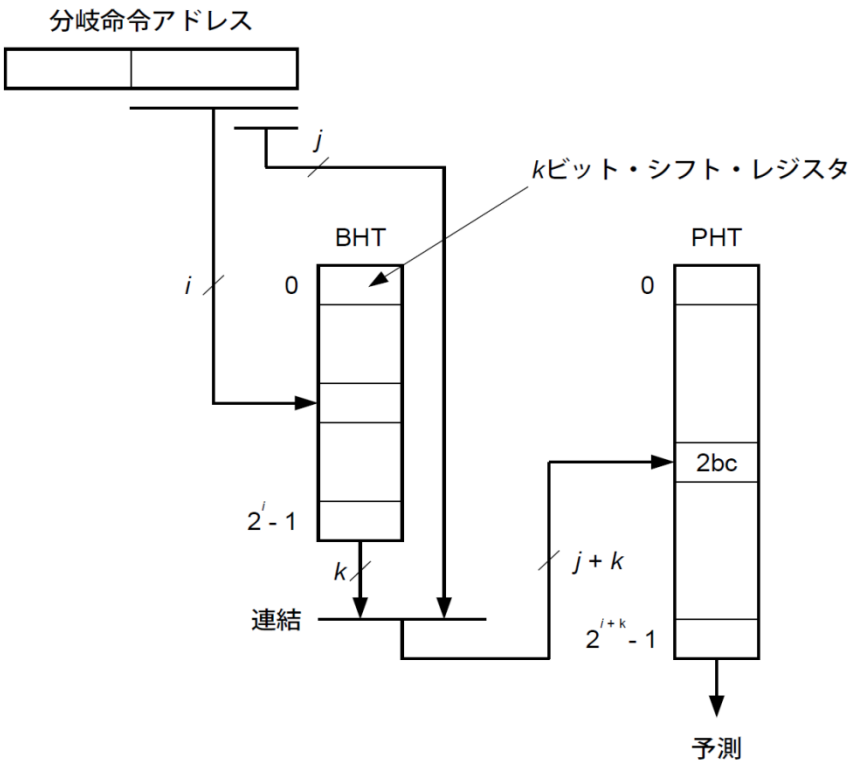
- PAs: Per-address Adaptive [Yeh 1993]



  - （分岐、ローカル分岐履歴）→2ビット・カウンタ
  - PHT:Pattern History Table

PAsの動作

- 分岐を実行したら、その結果(taken/not-taken)を、対応するエントリのBHTにアペンドする
- BHTの対応するエントリの値と分岐命令アドレスの一部を結合し、PHTのインデクスとする
- PHTを参照し、2以上ならtaken、1以下ならnot-taken



2bc vs. PAs

- 例：4回繰り返すループの分岐
  - TTN→T
  - TNT→T
  - NTT→T
  - TTT→N

パターン	2bc	PAs
TTN	T (正)	T (正)
TNT	T (正)	T (正)
NTT	T (正)	T (正)
TTT	T (誤)	N (正)

履歴ビット長k

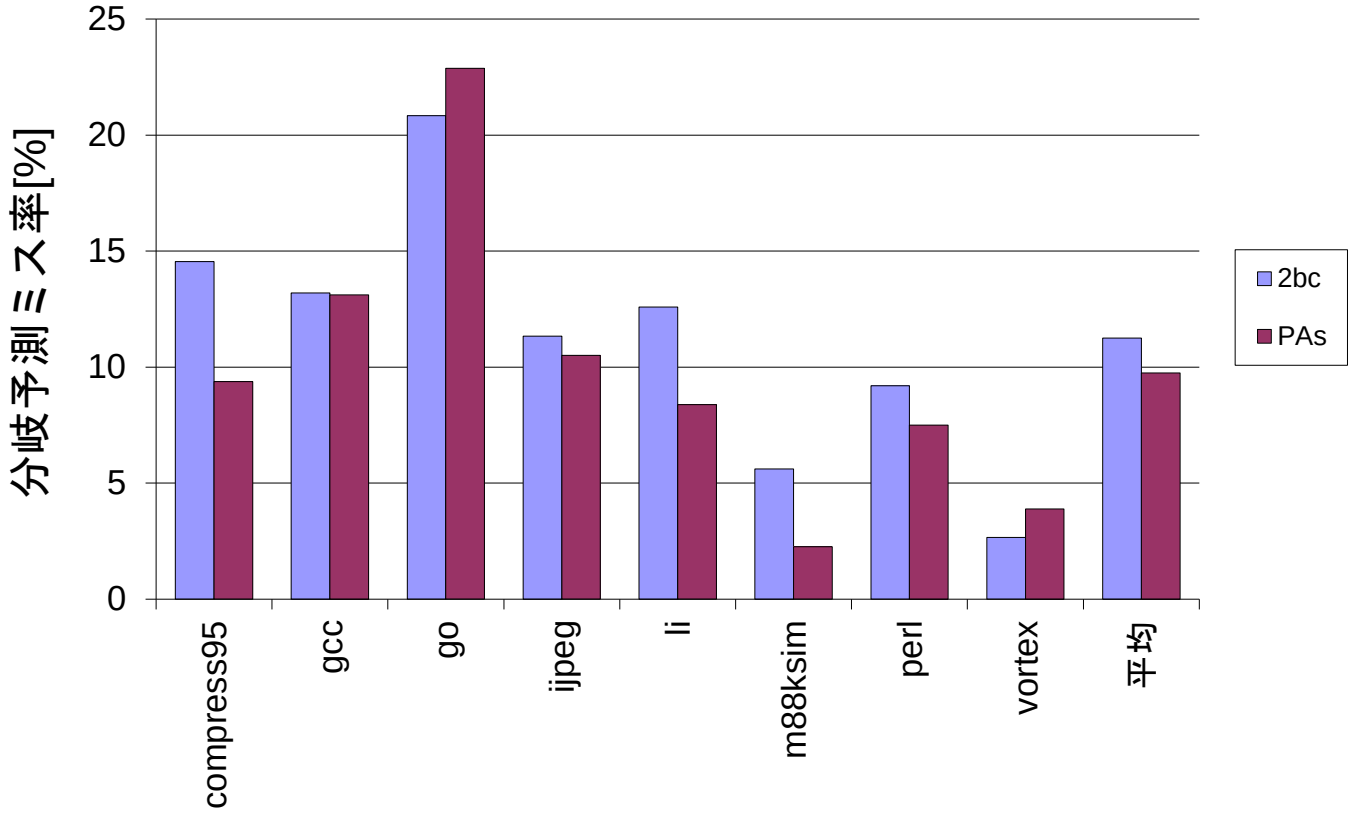
- k（BHTのエントリの長さ）と同じか短いパターンはとらえられる
  - パターン長がkより短いと、冗長（なだけ）

- kより長いパターンはとらえられない
  - ループ分岐のパターンの検出が主目的
  - 長いパターン

    →ほとんどtaken

    →問題は小さい

PAs予測精度



- コスト：0.5Kバイト（2bc:11bits, PAs(*i*,*j*,*k*)=(11,7,3)
- PAsは2bcより非常に優れている

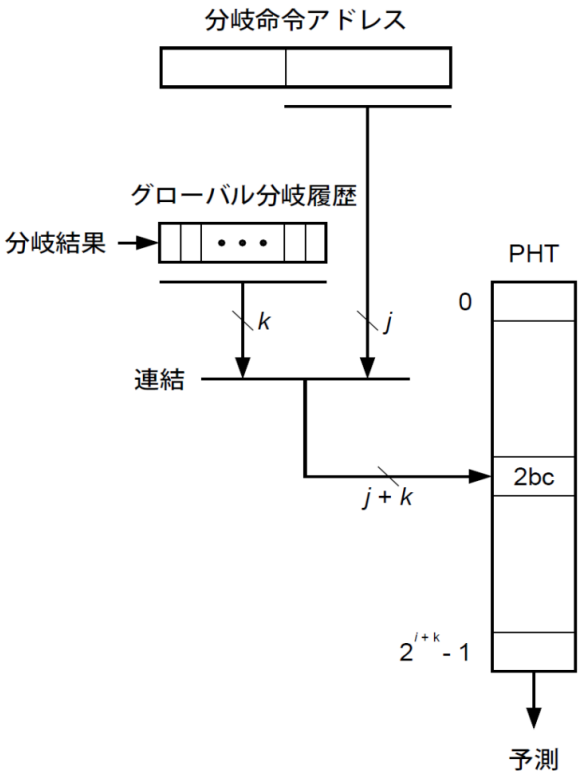
グローバル履歴2レベル適応型分岐予測

- 他の分岐と相関

```
b1: if (aa == 2) aa=0;
b2: if (bb == 2) bb=0;
b3: if (aa != bb) ...;
```

- パターン
  - TT→?
  - TN→?
- プログラムには意味があるため、分岐間には結果に相関がある

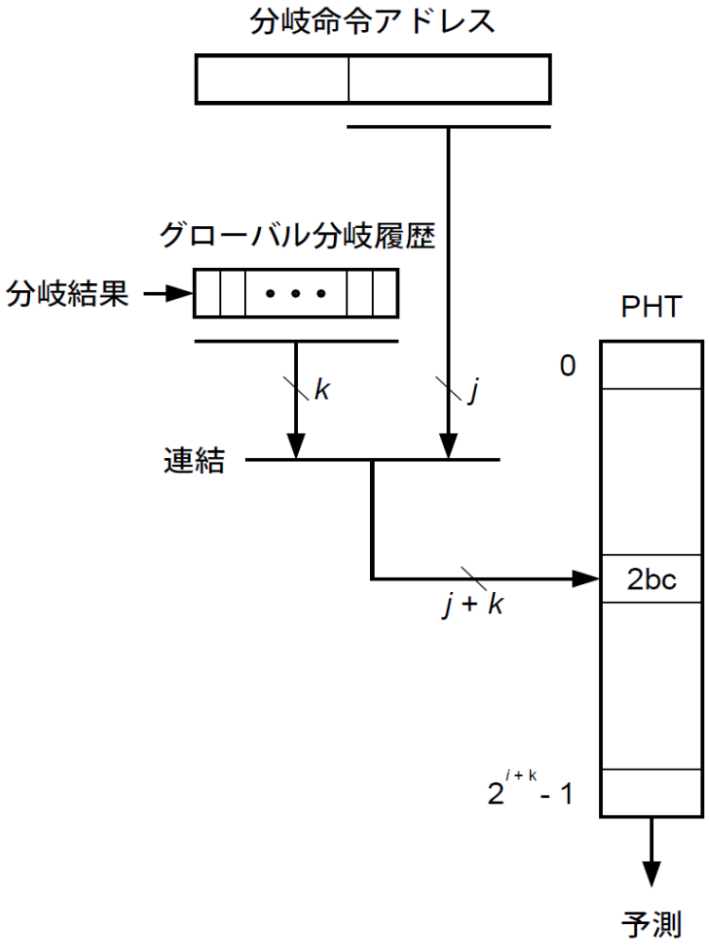
GAs: Global Adaptive [Pan 1992]



- （分岐、グローバル分岐履歴）→2ビット・カウンタ

GAsの動作

- 分岐命令を実行したら、結果(taken/not-taken)を、グローバル分岐履歴にアペンドする
- グローバル分岐履歴と分岐命令アドレスの一部を連結して、PHTのインデクスとする
- PHTを参照し、2以上ならtaken、1以下ならnot-taken



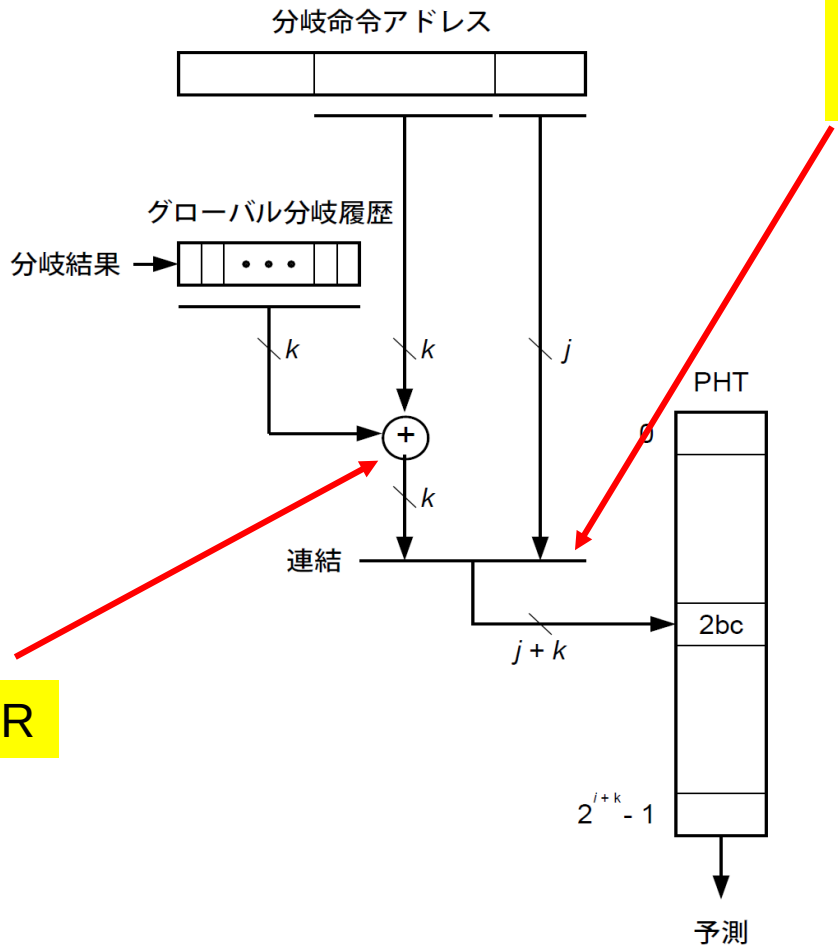
グローバル分岐履歴

- 長いほど予測精度良い
  - 遠い分岐にも相関
  - ローカル履歴に対する相関も利用可能

```
for(...){
   if()
}
```

- 長い分岐履歴によるコスト増加
  - （分岐アドレス*concatenate* グローバル分岐履歴）→2ビット・カウンタ
    - PHTコスト = 2^{j+k}
    - 無駄が多い
      ←グローバル分岐履歴によらず結果が一定（非常に偏りがある分岐）
  - ハッシング関数を変えて、高効率化
  - （分岐アドレス *XOR* グローバル分岐履歴）→2ビット・カウンタ
    - PHTコスト=2^j, $k<j$

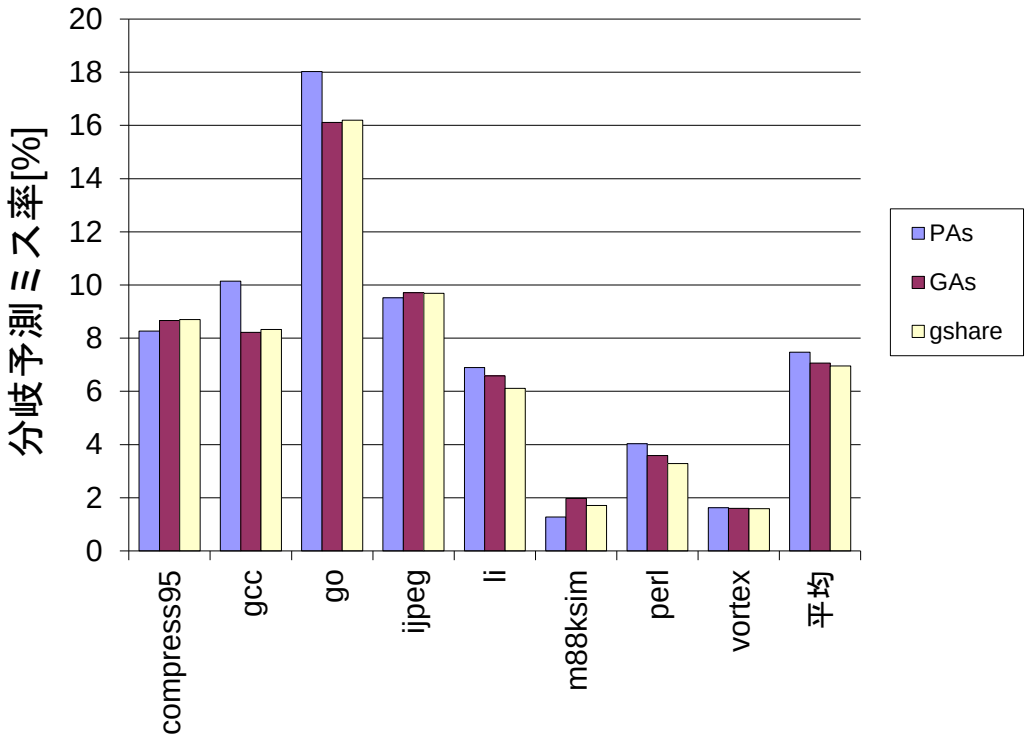
gshare [McFarling 1993]



競合を緩和するために
アドレスを連結する

「連結」ではなく XOR

GAsとgshareの予測精度(8KB)



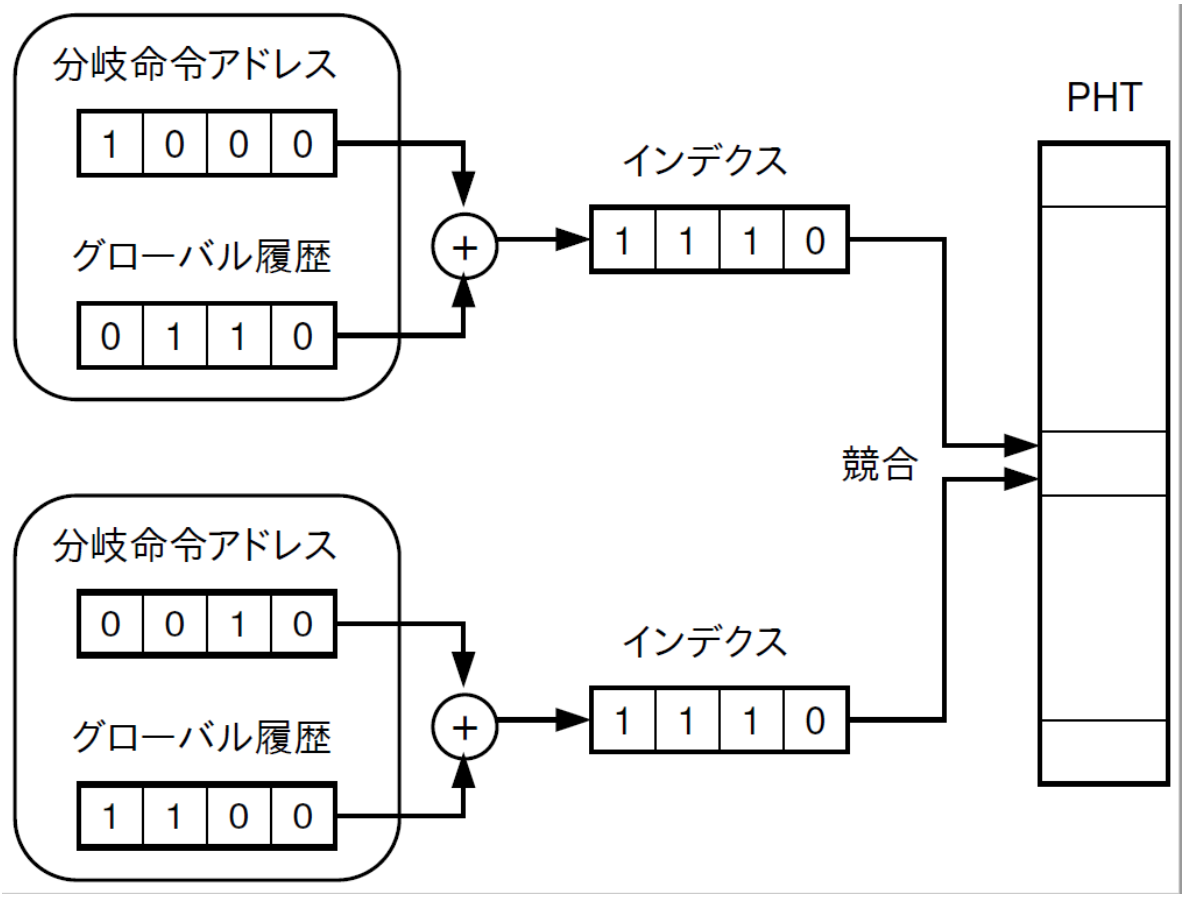
- GAs, gshareはPAsより優れている
- gshareはGAsより優れている

PHTでの競合

- 異なる（分岐アドレス、履歴）の組が同一のPHTエントリにマップされる場合がある

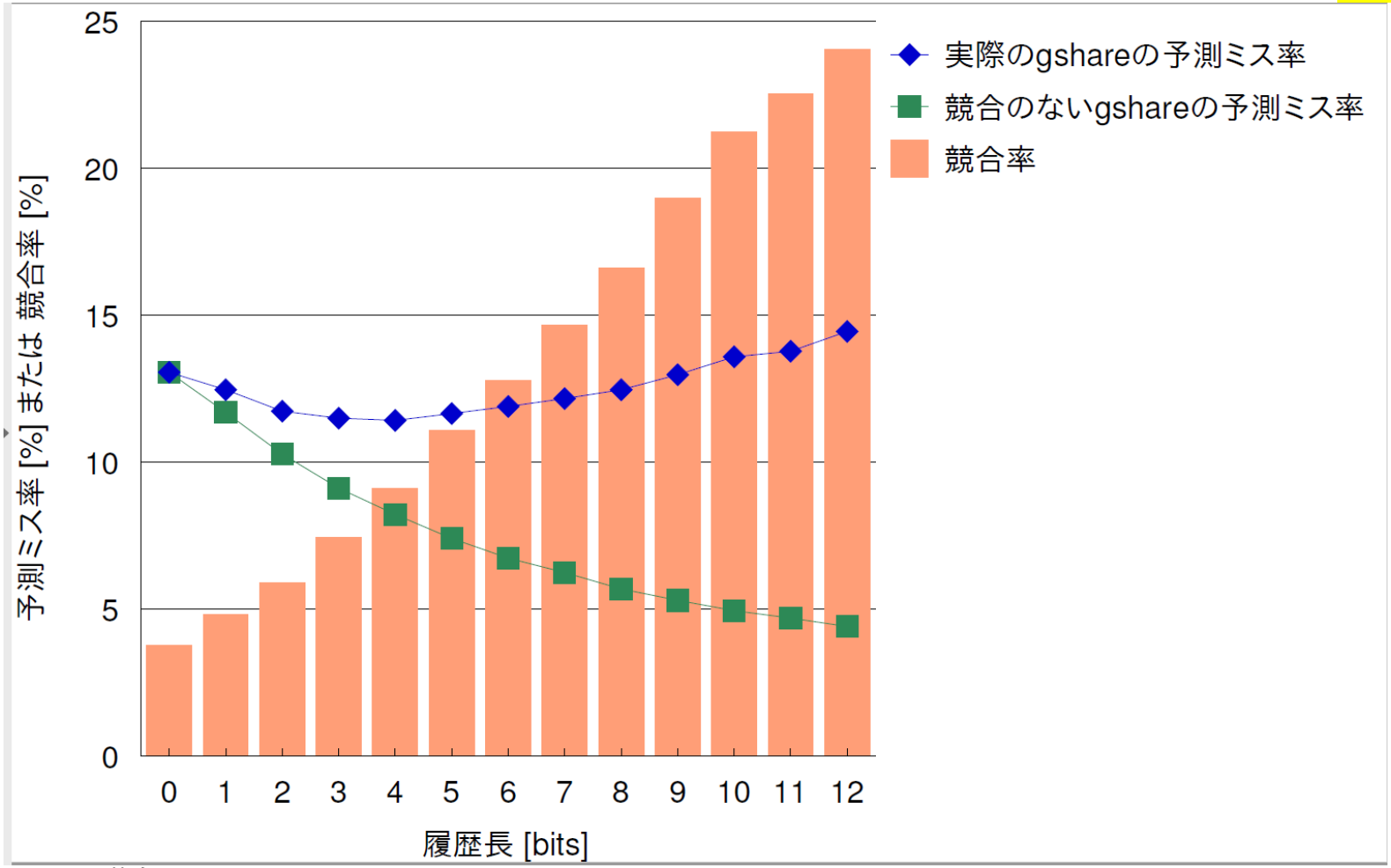
- 履歴が長いほど
  - 予測精度はあがるはず
  - 競合で損なわれる

PHTでの競合の例


分岐命令アドレス
1 0 0 0
グローバル履歴
0 1 1 0
+
インデクス
1 1 1 0
PHT
競合
分岐命令アドレス
0 0 1 0
グローバル履歴
1 1 0 0
+
インデクス
1 1 1 0

履歴長 vs. 予測精度、競合率

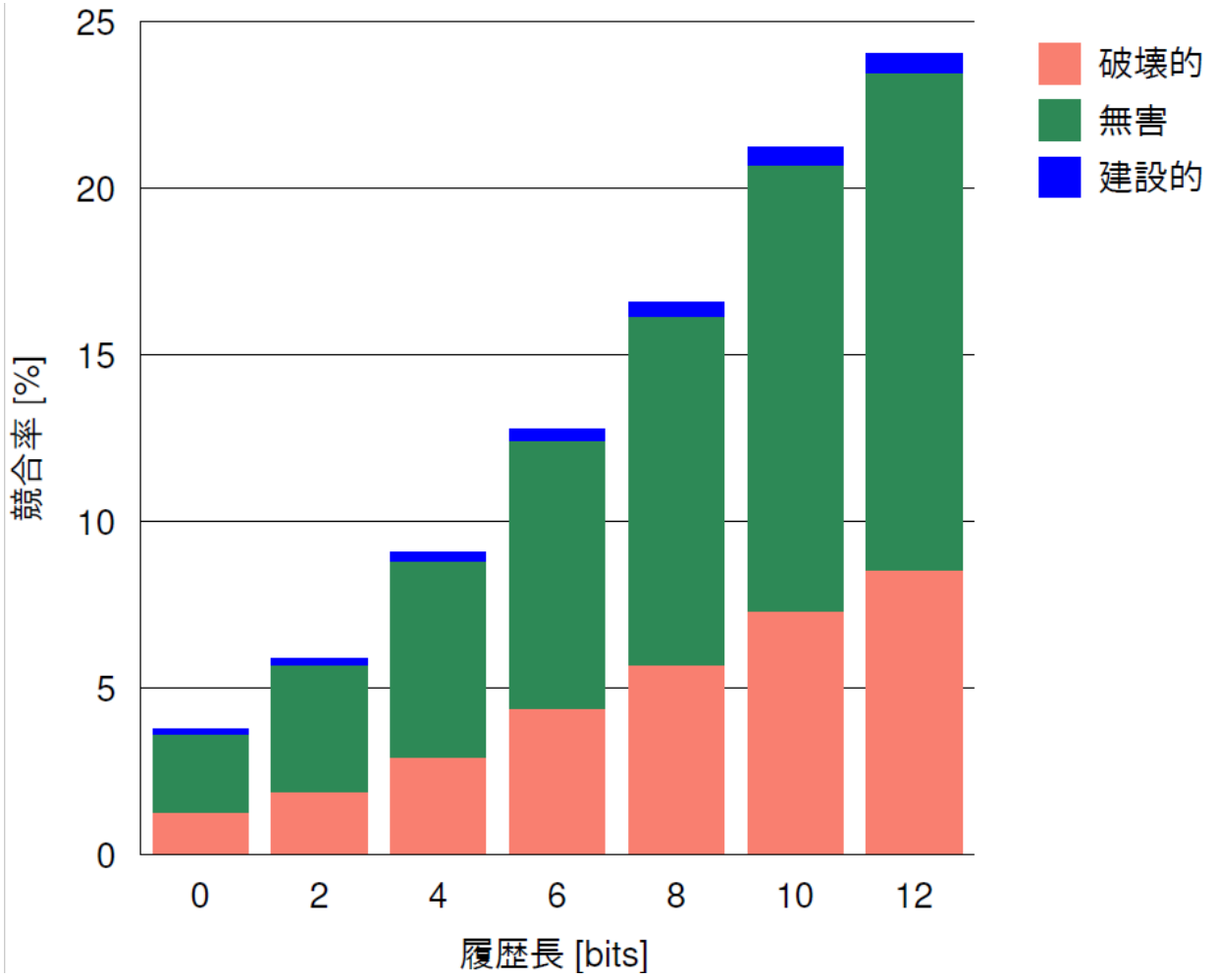
gcc



競合の性質 [Young 1995]

- 破壊的競合：競合が原因で予測を誤る
- 無害な競合：競合しても予測は同じ
- 建設的競合：競合が原因で予測が正しくなる

競合の内訳



競合による予測精度低下の抑制

- PHTを大きくする
  - コスト増大
  - 電力増大

フィルタリング [Chang 1996]

- 予測が困難な分岐のみgshareで予測
  - 容易な分岐は簡素な予測器（e.g., 1bc）で予測
  - 少数の分岐だけがPHTを使う→競合が減る

- 予測信頼度推定器[Jacobsen 1996]
  - 1bc予測が正しければ、カウンタ++
  - 誤りであれば、カウンタ・リセット
  - カウンタが飽和していれば、1bc予測は信頼できる

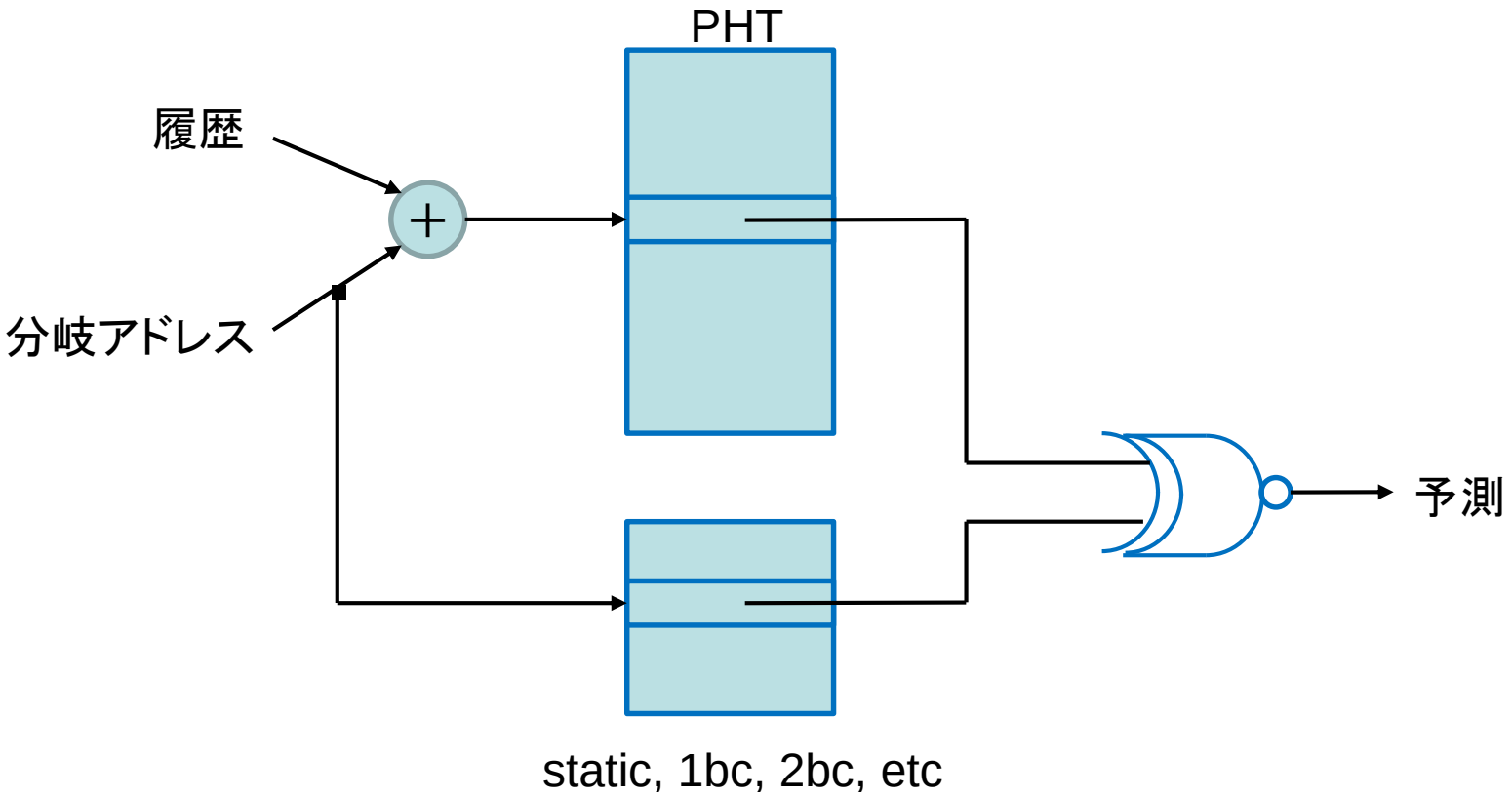
Agreeモード予測器 [Sprangle 1997]

- PHTのエンコードを変え、破壊的競合を無害な競合に変換

- 分岐ごとに分岐方向の偏りに、「同意するか？」を記録
  - 多くの分岐は偏っているので、同意が多い
  - 同意同士が競合しても、無害

- 偏り検出：簡素な予測器
  - 静的分岐予測
  - 1bc, 2bc

Agreeモード予測器の構成



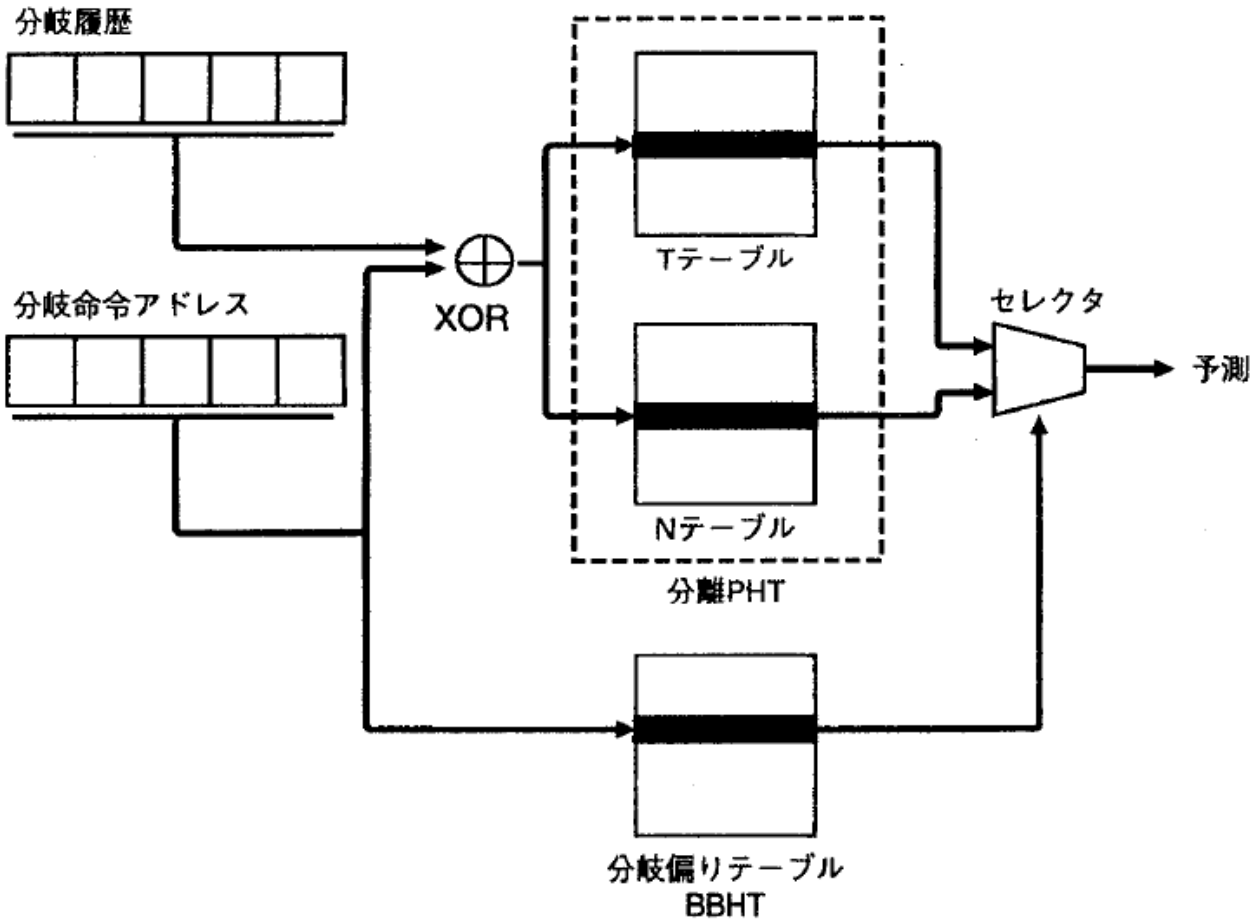
Bimode予測器 [Lee 1997]

- 競合の無害化

- takenに偏った分岐用のPHTと、not-takenに偏った分岐用のPHTに分ける
  - 偏りに応じて、一方のPHTのみ更新
  - taken同士、not-taken同士が競合しても無害

- 偏り検出
  - 簡素な予測器 (e.g., 2bc)

Bimode予測器の構成



Gskew予測器 [Michaud 1997]

- PHT: ハッシュ表
- 異なるハッシュ関数でマップする複数のPHT
  - 異なる競合が生じる
- 多数決

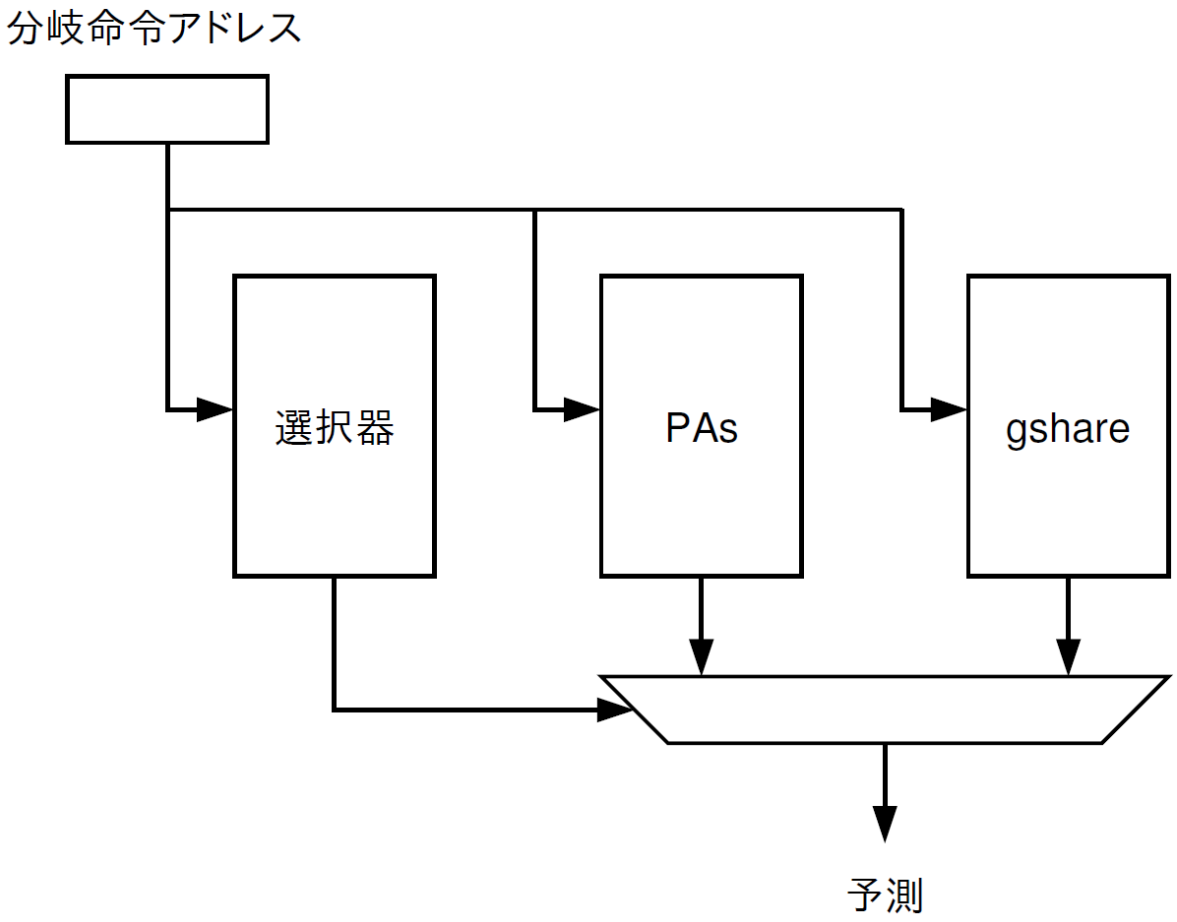
ハイブリッド予測器
[McFarling 1993, Chang 1995, Evers 1996]

- ローカル履歴に相関が強いか、グローバル履歴に相関が強いか？
  - 分岐ごとに異なる

- ローカル履歴予測器とグローバル履歴予測器を用意し、相関の強い方にマップする

- 選択器：どちらが正解を出したかの頻度を記録するU/Dカウンタ
  - (P1,P2) = (Y,N)→UP
  - (P1,P2) = (N,Y)→DOWN
  - (P1,P2) = (Y,Y) = (N,N)→そのまま

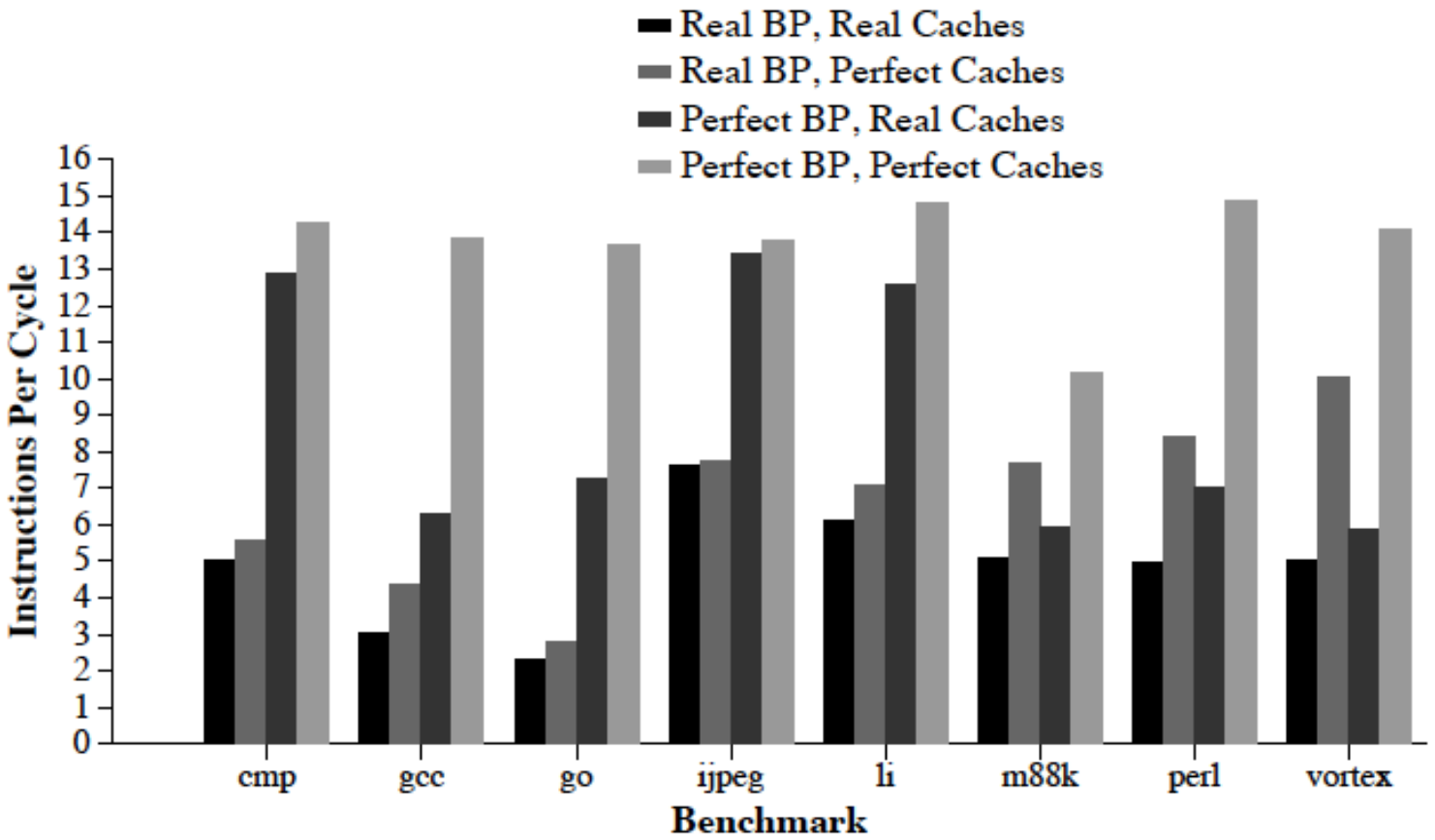
ハイブリッド予測器の構成



内容

- 目的
- 2ビット・カウンタ方式 [Lee 1984]（復習）
- 2レベル適応型方式
  - PAs [Yeh 1991, Yeh 1992, Yeh 1993]
  - GAs [Pan 1992]
  - gshare [McFarling 1993]
  - 競合の低減手法 [Chang 1996, Sprangle 1997, Lee 1997, Michaud 1997]
- ハイブリッド予測器 [McFarling 1993, Chang 1995, Evers 1996]
- 最新の予測器 TAGE [Seznec 2006]

分岐予測精度は十分か？

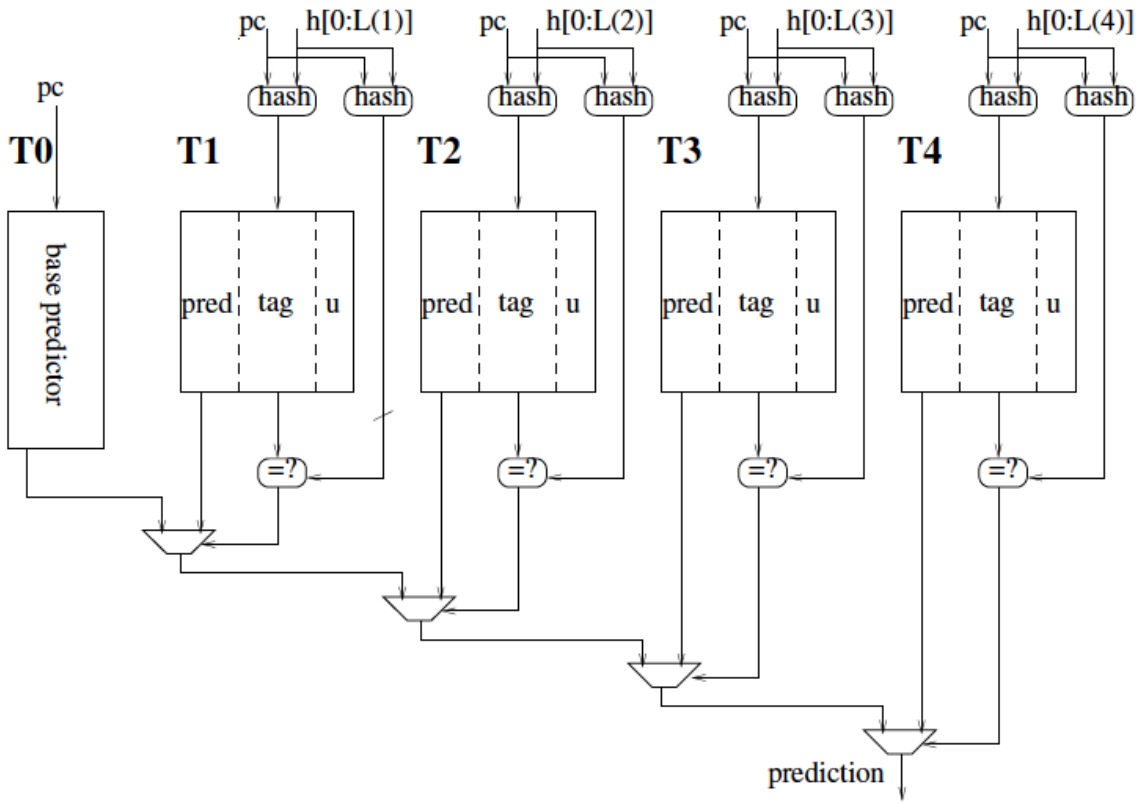


R. S. Chappell, J. Stark, S. P. Kim, S. K. Reinhardt, Yale N. Patt, Simultaneous subordinate microthreading (SSMT), In *Proceedings of the 26th Annual International Symposium on Computer architecture*, pp.186-195, May 1999.

TAGE予測器 [Seznec 2006]

- 現在最も優れた予測器の1つ
- 分岐によって、予測に最適な履歴長がある
  - 容易な分岐
    - 履歴は短くてよい
    - 長いとPHTを汚す
  - 難しい分岐
    - 履歴は長い必要がある
    - 履歴長100+
- PHTにタグ → 競合を検出

TAGE予測器の構成



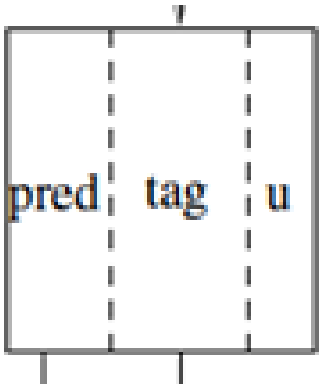
A. Seznec and P. Michaud, A case for (partially) TAgged GEometric history length branch prediction, *Journal of Instruction Level Parallelism*, Vol. 8, February 2006.

PHTs

- 種々の履歴長のPHT
  - e.g., 8 PHTs : {0, 2, 4, 8, ..., 128}
- タグあり
  - 競合検出
  - タグ不一致ならば、そのPHTでは予測をしない
- より長い履歴のPHTの予測を採用する
- 予測を誤ったら、長い履歴のPHTにも割り当てようとする
- 割り当ての有用度を推定し、最適な割り当てに使う

PHTのエントリ

- pred: 偏りを表す3ビットU/Dカウンタ
- タグ : PCと分岐履歴のハッシュ値
- u: エントリ割り当ての有用度を示す2ビットU/Dカウンタ



predカウンタの更新

- takenならアップ、not-takenならダウン

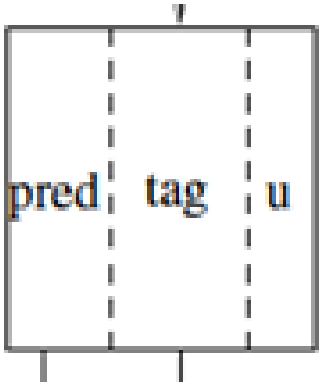
用語

- provider
  - 最終的な予測を出したPHT
  - タグ一致の中で、最も履歴の長いPHT
- altpred
  - もしもproviderがタグ不一致で予測をしないなら、予測をしたであろう第2の候補者

u: 有用度カウンタ

- 最適な割り当てに使う
- 他のPHTでは予測できない予測をし、それが正しければ、有用度向上

```
if (provider's pred != altpred's pred)
   if (correct pred)
      u++;
   else
      u--;
else
   // neutral, do nothing
```



- 古くなったら有用性低下←時間的局所性がある
  - 定期的に（256K分岐）MSBをリセット、次（？）にはLSBをリセット

予測誤り時

- providerのカウンタを更新

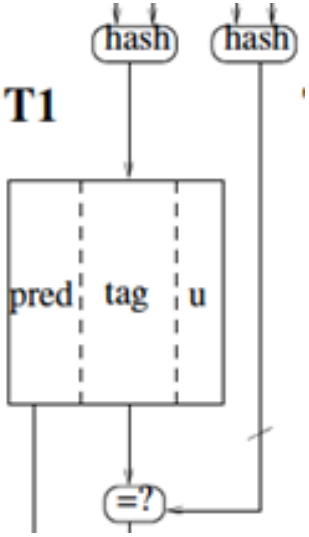
- より履歴の長いPHTにも割り当てようとする

```
if (k exists, u_k = 0 (i < k < M))
   allocate T_k (for minimum k)
else
   foreach j (i < j < M)
      u_j--;
      // no allocation
```

- 最後の foreach のところ
  - 履歴の長いPHTのエントリに割り当てやすくする

ハッシュの作り方 [Michaud 2005]

- 履歴長 > PHTインデクス
  - h[0:9] ^ h[10:19] ^ h[20:29] ^ h[30:39]
  - $\log_2 N$段の遅延

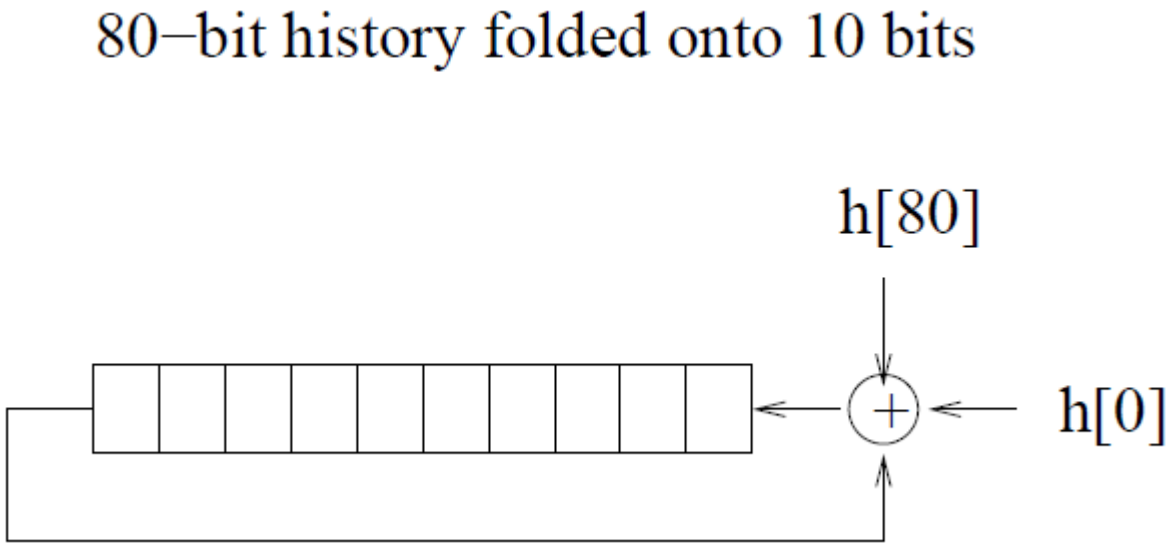


- 循環シフトレジスタ(CSR)に履歴を畳み込み

- PC[9:0] ^ PC[19:10] ^ CSR1 ^ CSR2 ^ CSR3
  - 複数のCSRを使って、CSRが周期的パターンになるのを防ぐ

P. Michaud, A PPM-like, Tag-based Predictor, *Journal of Instruction-Level Parallelism*, Vol.7, April 2005.

CSRの例



まとめ

- 分岐予測
  - 深いパイプラインにおける制御ハザード→大きな性能低下
  - 過去の履歴から将来の分岐の振る舞いを予測
  - 2ビット・カウンタ方式

- 2レベル適応型分岐予測
  - パターンをとらえて予測
  - ローカル履歴方式
    - 自身のパターンとの相関
  - グローバル履歴方式
    - 他の分岐のパターンとの相関
  - 破壊的競合を削減する手法
  - 最適な履歴長に応じたPHT