

**Universidade Federal do Rio de Janeiro
Pós-Graduação em Informática IM-NCE/UFRJ**

Microarquitecturas de Alto Desempenho

Predição de Desvio

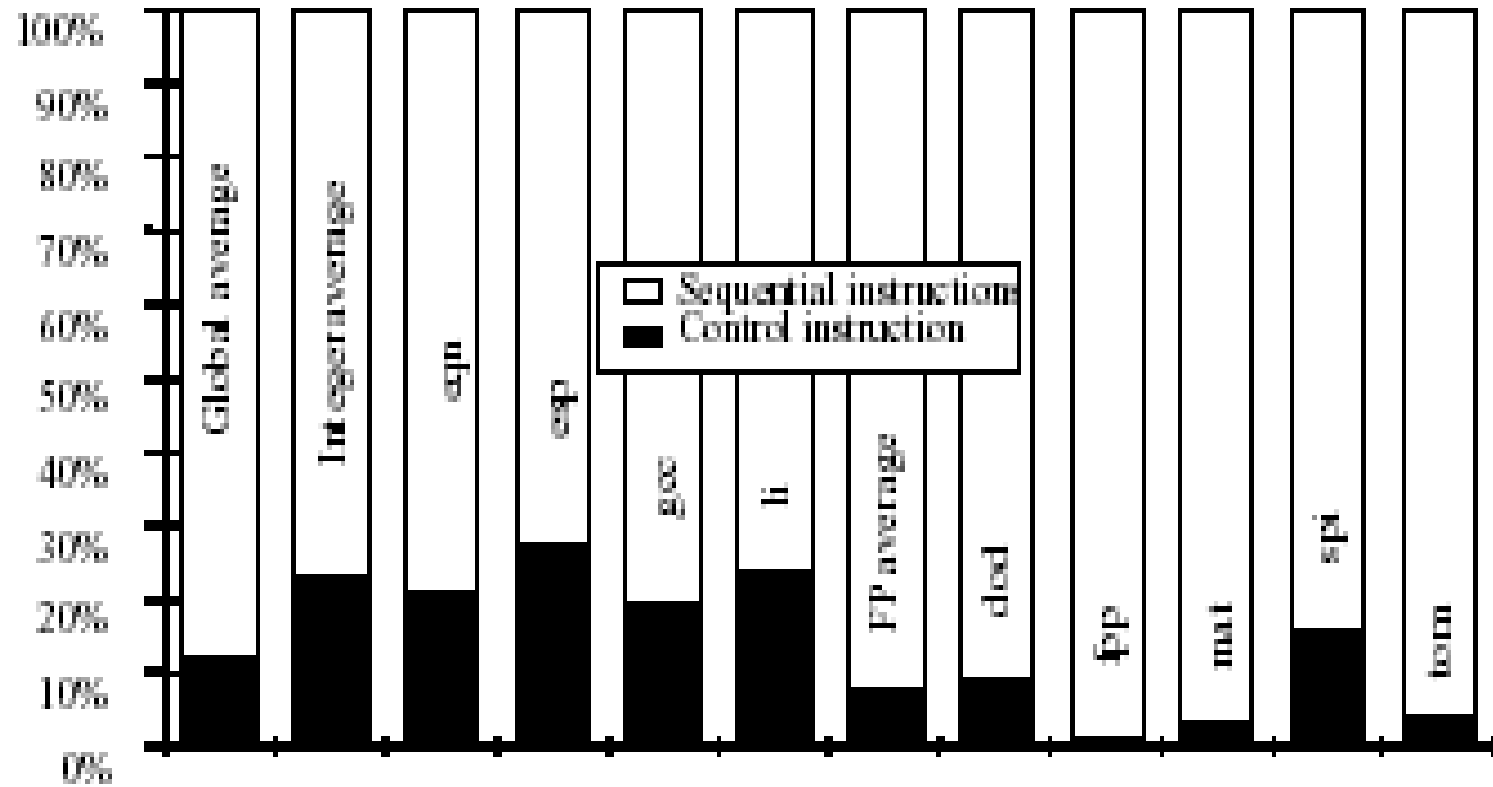
Gabriel P. Silva

Introdução

- **Desvios:**
 - **Instruções que podem alterar o fluxo de execução das instruções de um programa.**

	Condicional	Incondicional
Direto	if - then- else for loops (bez, bnez, etc)	procedure calls (jal) goto (j)
Indireto		return (jr) virtual function lookup function pointers (jalr)

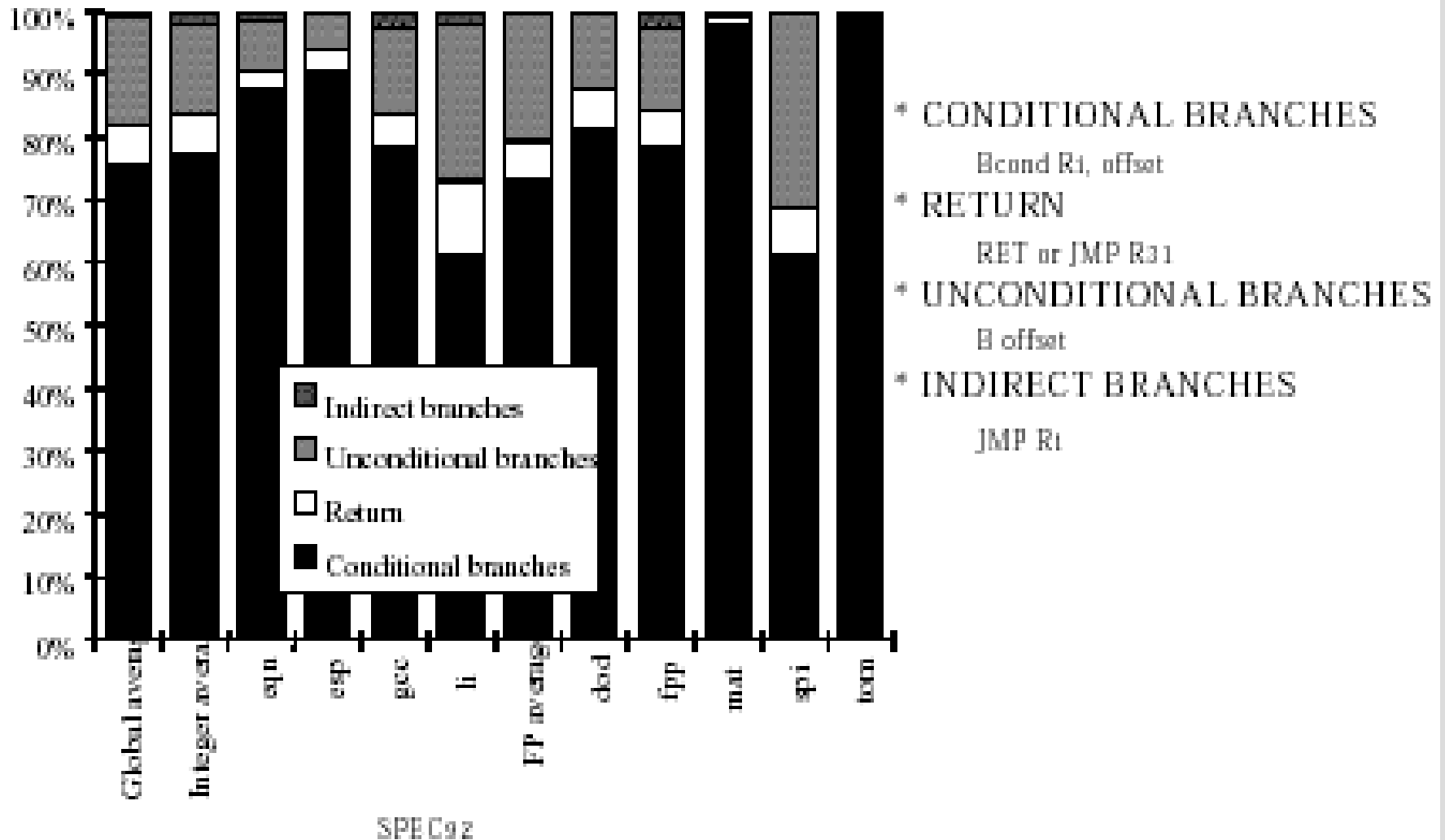
Introdução



SPEC92

Yeh and Patt

Introdução



Há necessidade de predição?

- **Predição de desvio:**
 - **Aumenta o número de instruções disponíveis para o despacho;**
 - **Aumenta o paralelismo a nível de instrução;**
 - **Permite que trabalho útil seja concluído enquanto se espera pela resolução do desvio.**

Predição de Desvio

- **Predizer o resultado de um desvio**
 - **Direção:**
 - Tomado / Não Tomado
 - Preditores de Direção
 - **Endereço Alvo:**
 - Tomado: PC+offset
 - Não Tomado: PC+4
 - Preditores de Endereço Alvo
 - Branch Target Address Cache (BTAC) ou Branch Target Buffer (BTB)

Conseqüências da Predição

- **Nenhum estado especulativo pode concluir, ou seja, nem a memória, nem os registradores podem ser modificados;**
- **Há necessidade de tratar as exceções corretamente, ou seja, instruções especulativas não podem gerar exceção;**
- **As instruções especulativas se acumulam no pipeline, aguardando a resolução do desvio.**

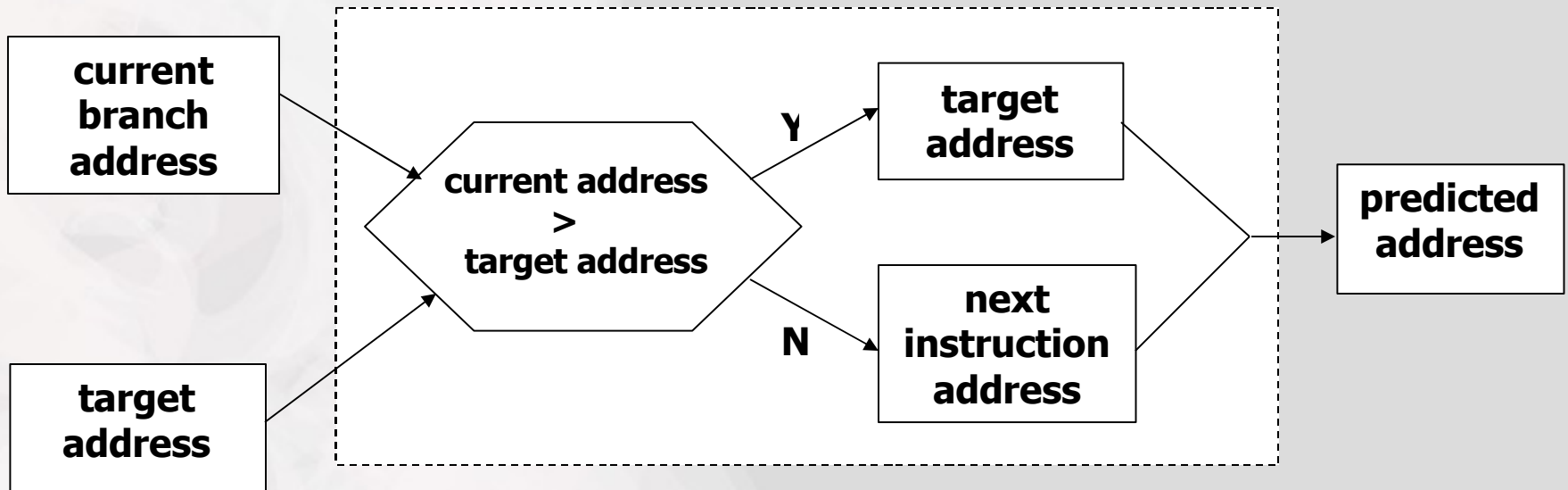
Estratégias de Predição de Desvios

- As estratégias de predição de desvio podem ser divididas em duas categorias básicas:
 - Estratégia de predição de desvio estática por meio de mecanismos de “software”;
 - Decidida a priori pelo compilador
 - Estratégia de predição de desvio dinâmica por meio de mecanismos de “hardware”.
 - As decisões de predição podem ser alteradas durante a execução do programa

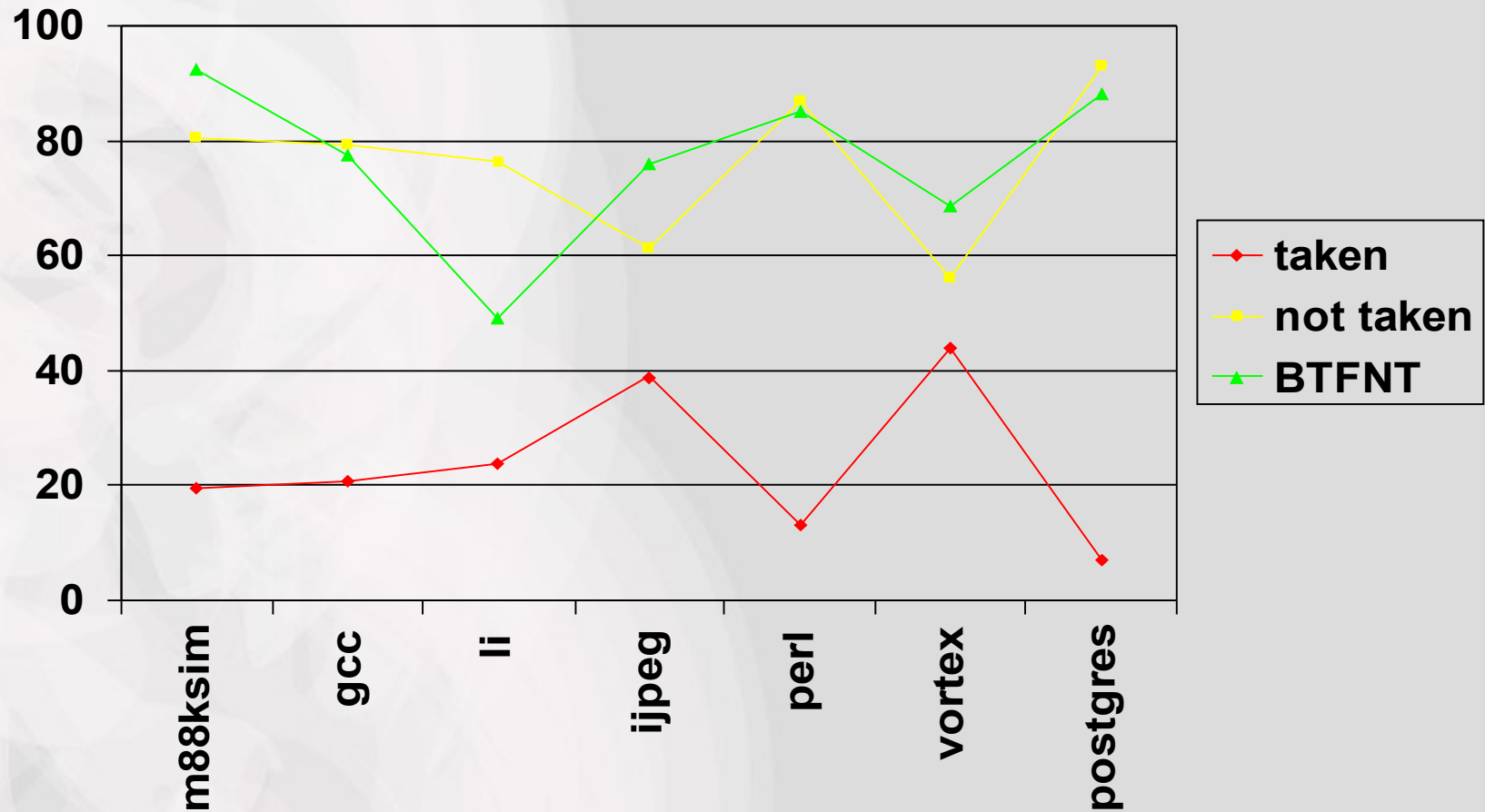
Predição Estática

- As estratégias mais utilizadas são:
 1. **Taken:** Predizer que todos os desvios serão tomados.
 2. **Not-taken:** Predizer que todos os desvios **não** serão tomados.
 3. **Back-taken:** Predizer que todos os desvios para **trás** serão **tomados** e que todos os desvios para **frente não** serão tomados.
 4. Predizer que todos os desvios com certos códigos de operação serão tomados e que os demais não serão tomados.
 5. Baseadas em profile

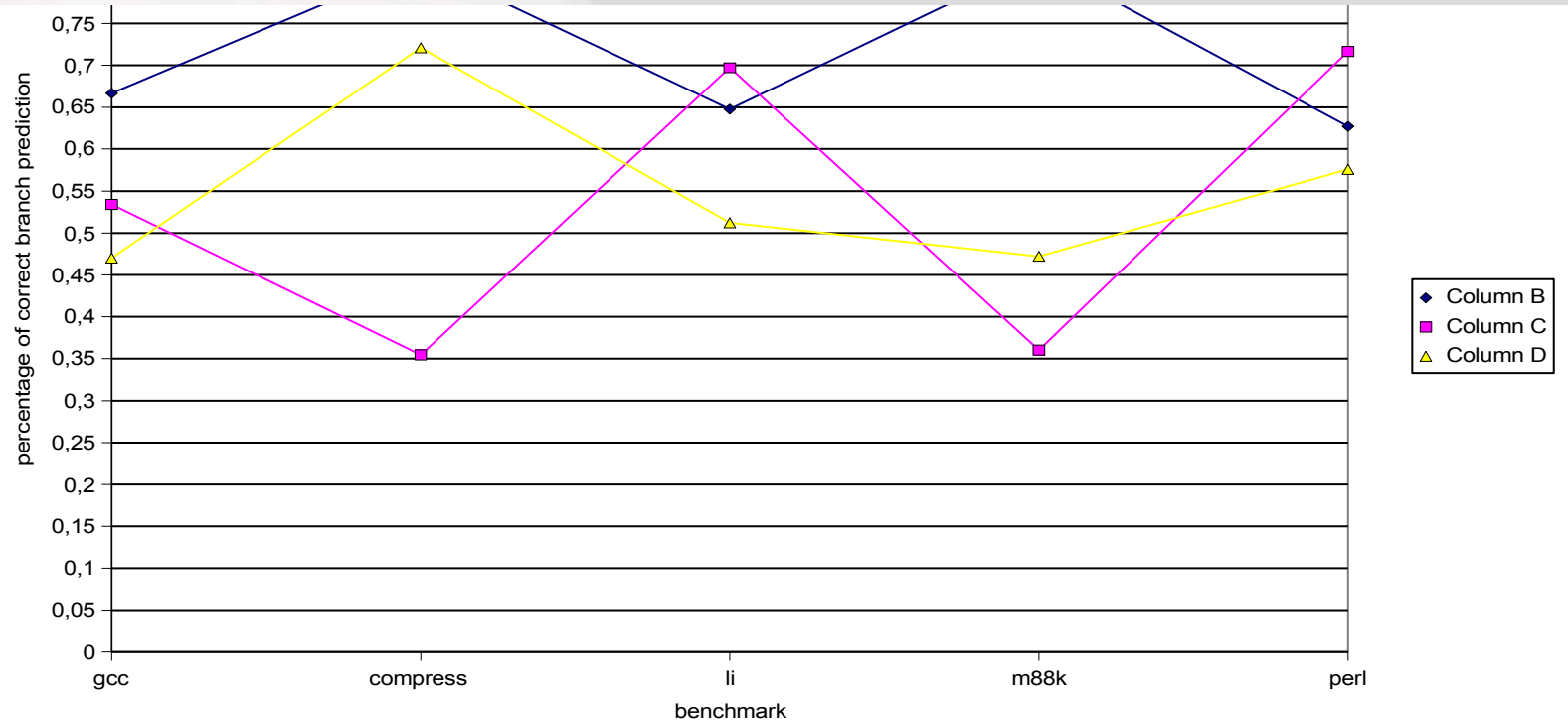
Backward Taken



Preditores de Desvio Estáticos



Preditores de Desvio Estáticos



Predição Dinâmica

- As estratégias de predição dinâmica mais utilizadas são:

- 1. Predição com um bit**

- 2. Predição Bimodal em um Nível**

- 3. Predição Correlacionada (Dois Níveis)**

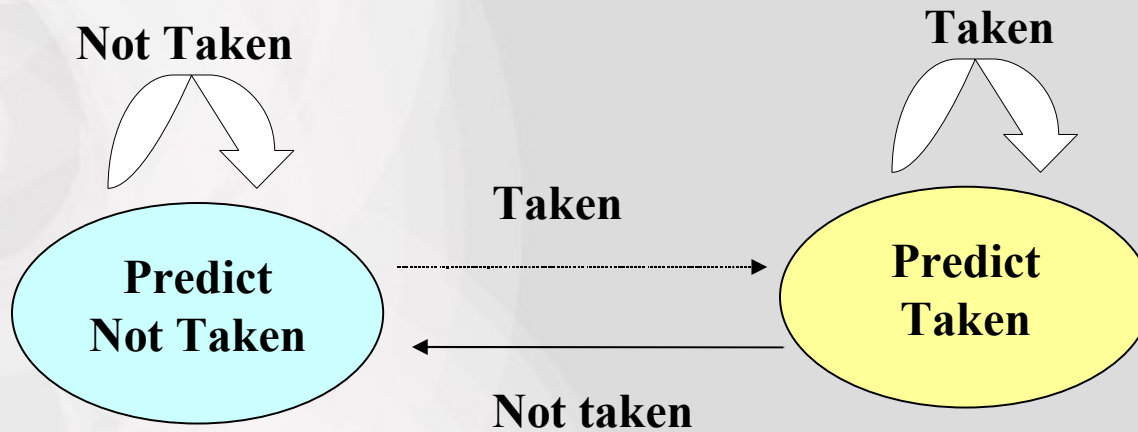
- **GAg**
- **GShare**
- **Gselect**
- **GAp**
- **PAp**

- 4. Tabela de Instruções de Desvio**

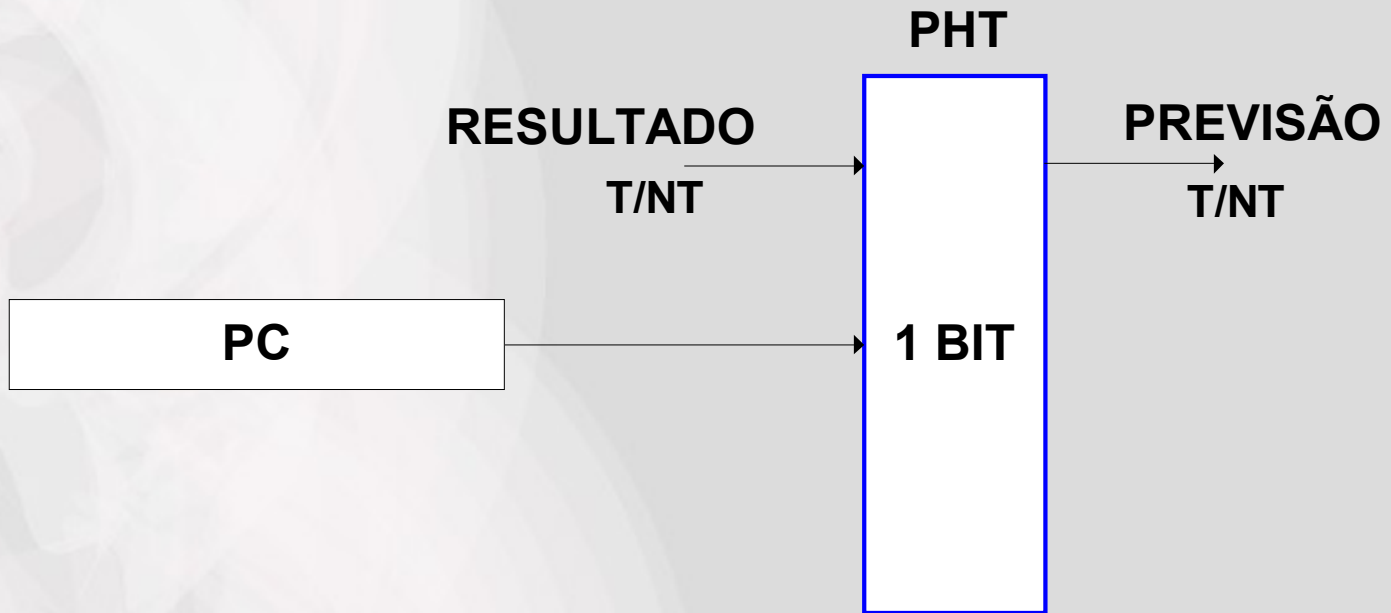
Predição com 1 Bit

- **Utiliza-se uma tabela de histórico de predição (PHT) que é endereçada com a parte mais baixa do endereço da instrução de desvio.**
- **Cada posição da tabela contém um bit que diz se o desvio foi tomado ou não da última vez em que foi executado, que servirá como previsão para a instrução de desvio atual.**
- **Se a predição for errada, o bit de predição será invertido e armazenado na PHT.**
- **Se um desvio é quase sempre tomado, é muito provável que ele seja predito incorretamente duas vezes, ao invés de uma, quando não for tomado.**

Predição com 1 Bit



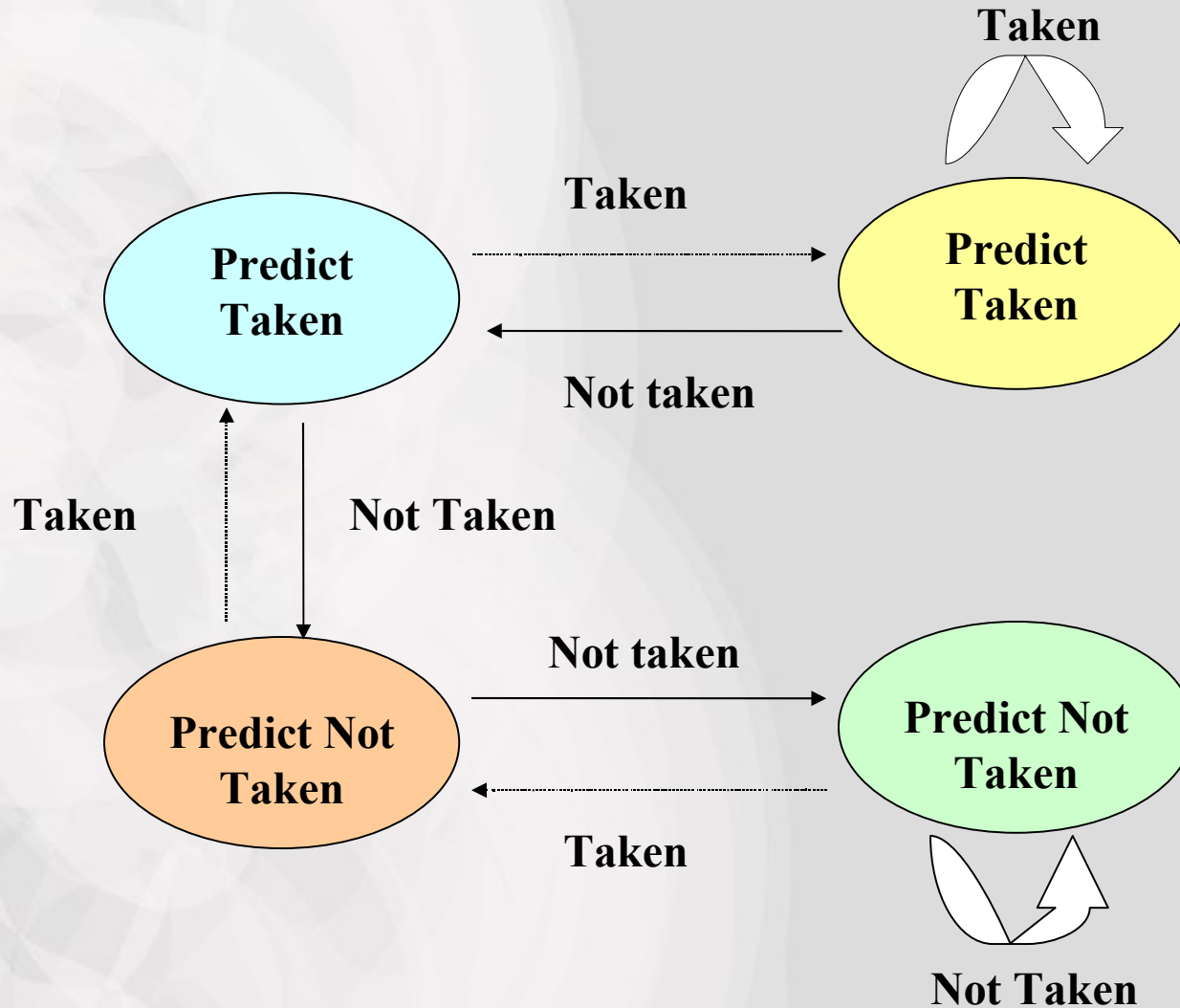
Predição com 1 Bit



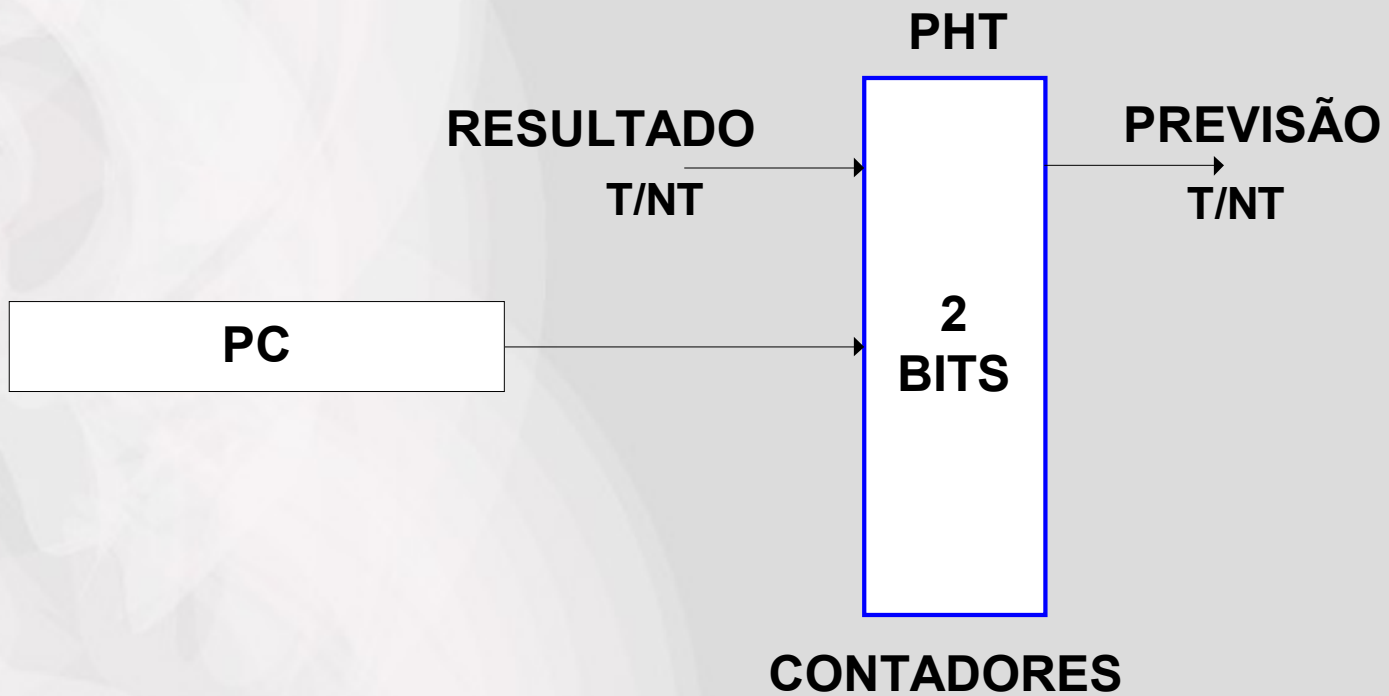
Predição Bimodal

- O preditor bimodal é essencialmente um contador saturante de dois bits que assume valores entre 0 e 3.
- Quando o contador é maior ou igual a 2, o desvio é predito com tomado; em caso contrário, como não tomado.
- O contador é incrementado cada vez que o desvio é tomado e decrementado cada vez que o desvio não é tomado.

Predição Bimodal

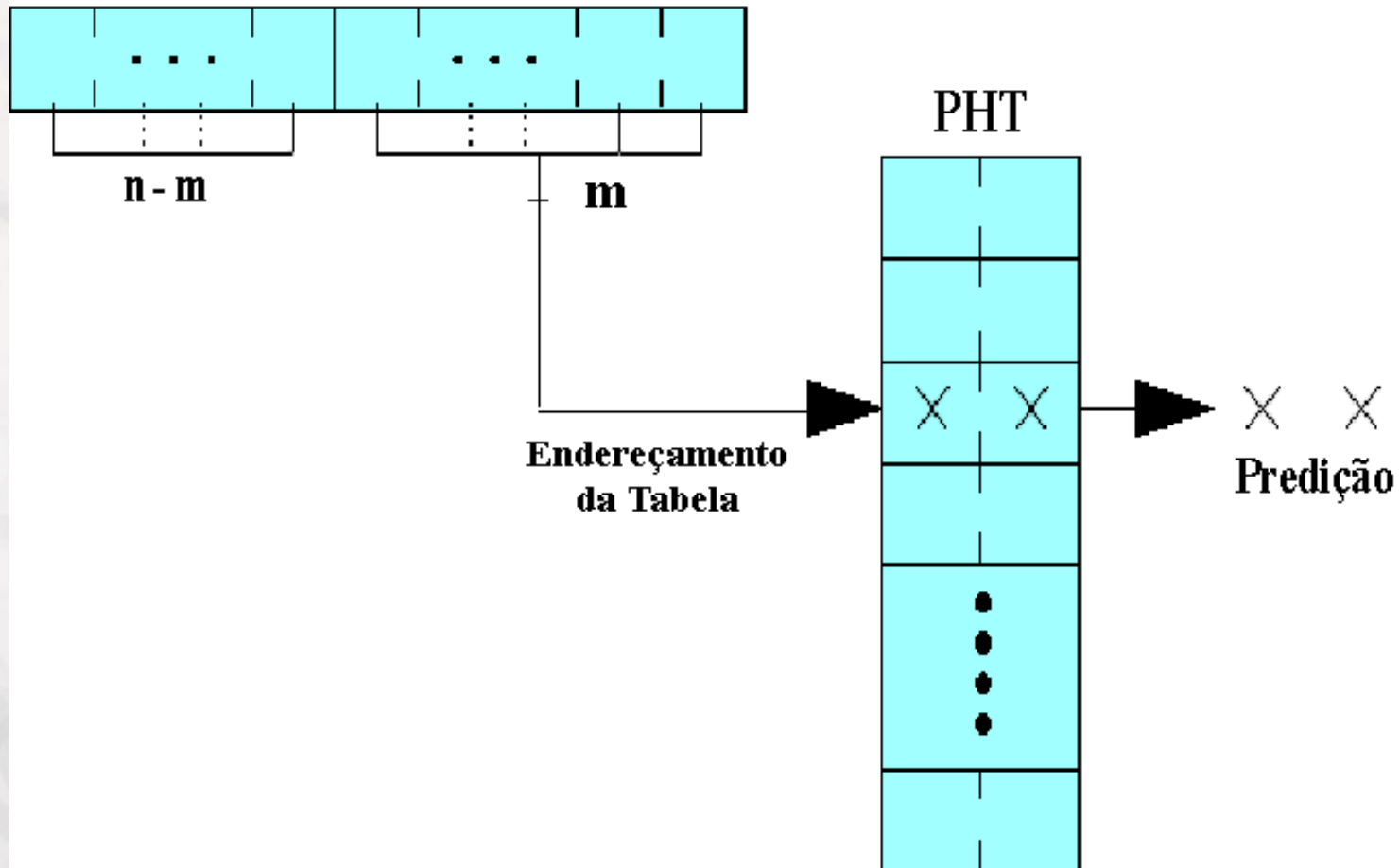


Predição Bimodal



Predição Bimodal

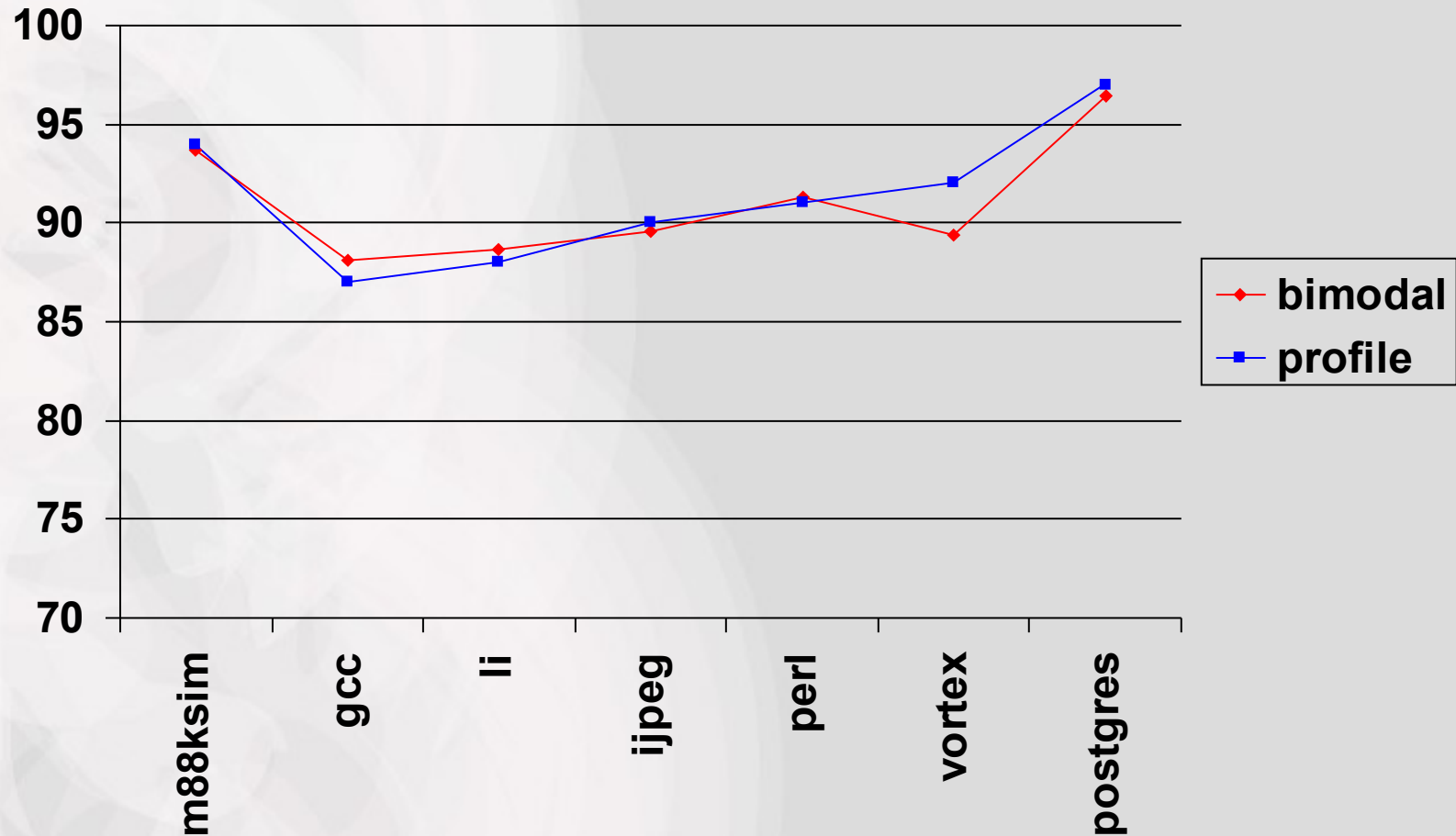
Endereço Atual do Desvio (n bits)



Preditores de 1-bit x 2-bits

- **Um preditor de 1-bit prediz de modo correto um desvio ao final de uma iteração de um loop, enquanto o loop não termina.**
- **Em loops aninhados, um preditor de 1-bit irá causar duas previsões incorretas para o loop interno:**
 - **Uma vez no final do loop, quando a iteração termina o loop ao invés de ir para o começo do loop, e**
 - **Uma vez quando a primeira iteração do loop for reiniciada, quando ele prediz o término do loop ao invés do começo do loop.**

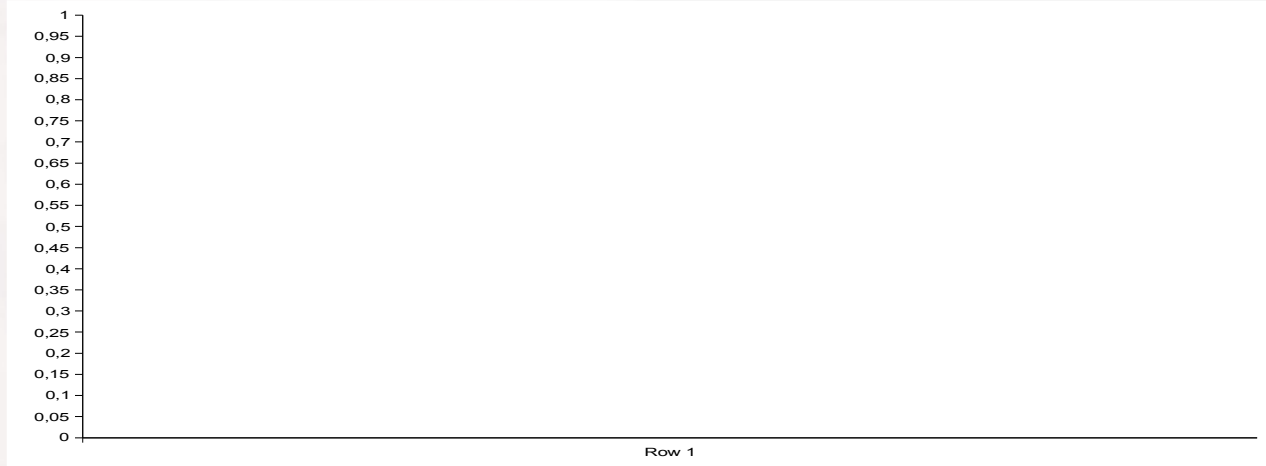
Preditores Bimodais



Preditores de 1-bit x 2-bits

- Este erro duplo em *loops* aninhados é evitado por um esquema de predição de dois bits.
- **Preditor de 2-bits:** Uma predição deve errar duas vezes antes de ser alterada quando um preditor de 2-bits é utilizado.
- Em loops aninhados, um preditor de bimodal irá causar apenas uma predição incorreta para o loop interno:
 - Uma vez no final do loop, quando a iteração termina o loop ao invés de ir para o começo do loop, e
- Quando a primeira iteração do loop for reiniciada, ele mantém a predição correta de ida para o começo do loop.

Preditores de 1-bit x 2-bits



Preditor por Correlação

- O preditor bimodal utiliza apenas o comportamento mais recente de um desvio para prever o comportamento futuro daquele desvio.
- O comportamento do desvio de instruções diferentes, contudo, freqüentemente estão mutuamente relacionados.
- É possível melhorar a acurácia da predição se olharmos também para o comportamento recente de outros desvios, ao invés de apenas aquele que estamos tentando prever.
- Este raciocínio levou ao desenvolvimento dos preditores correlacionados e aos de dois níveis.

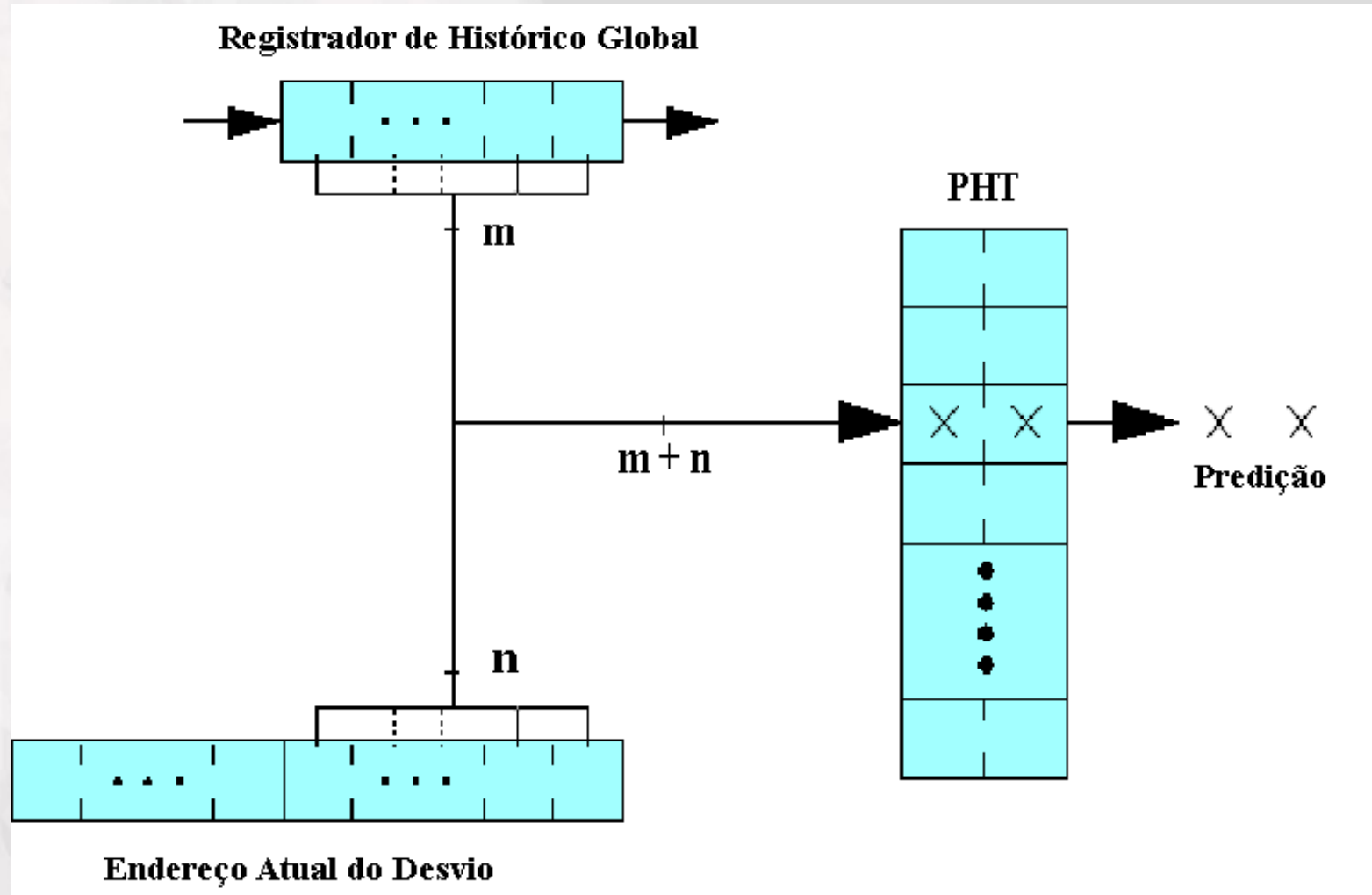
Preditor por Correlação

- Enquanto os **preditores bimodais** usam apenas o seu próprio histórico, os **preditores correlacionados** também usam o histórico dos desvios vizinhos.
- **Registrador de Histórico de Desvio (BHR)**: O histórico global dos m desvios mais recentes podem ser gravados em um registrador de m -bits, onde cada bit registra se o desvio foi tomado ou não.
- **Notação**: um **preditor (m,n)** usa o comportamento dos últimos m desvios para escolher entre 2^m preditores de desvios, cada qual é um preditor de n -bits para um único desvio.

Preditor Gselect

- **Concatena alguns bits de mais baixa ordem do endereço do desvio e os bits do registrador de histórico global (BHR).**
- **Assim, o acesso à PHT é feito levando em conta o comportamento dos últimos desvios executados, expresso nos bits do BHR.**
- **A relação entre os bits do endereço de desvio e os bits do BHR utilizados para endereçar a PHT, reforçam um caráter mais local ou global para a a predição a ser realizada.**

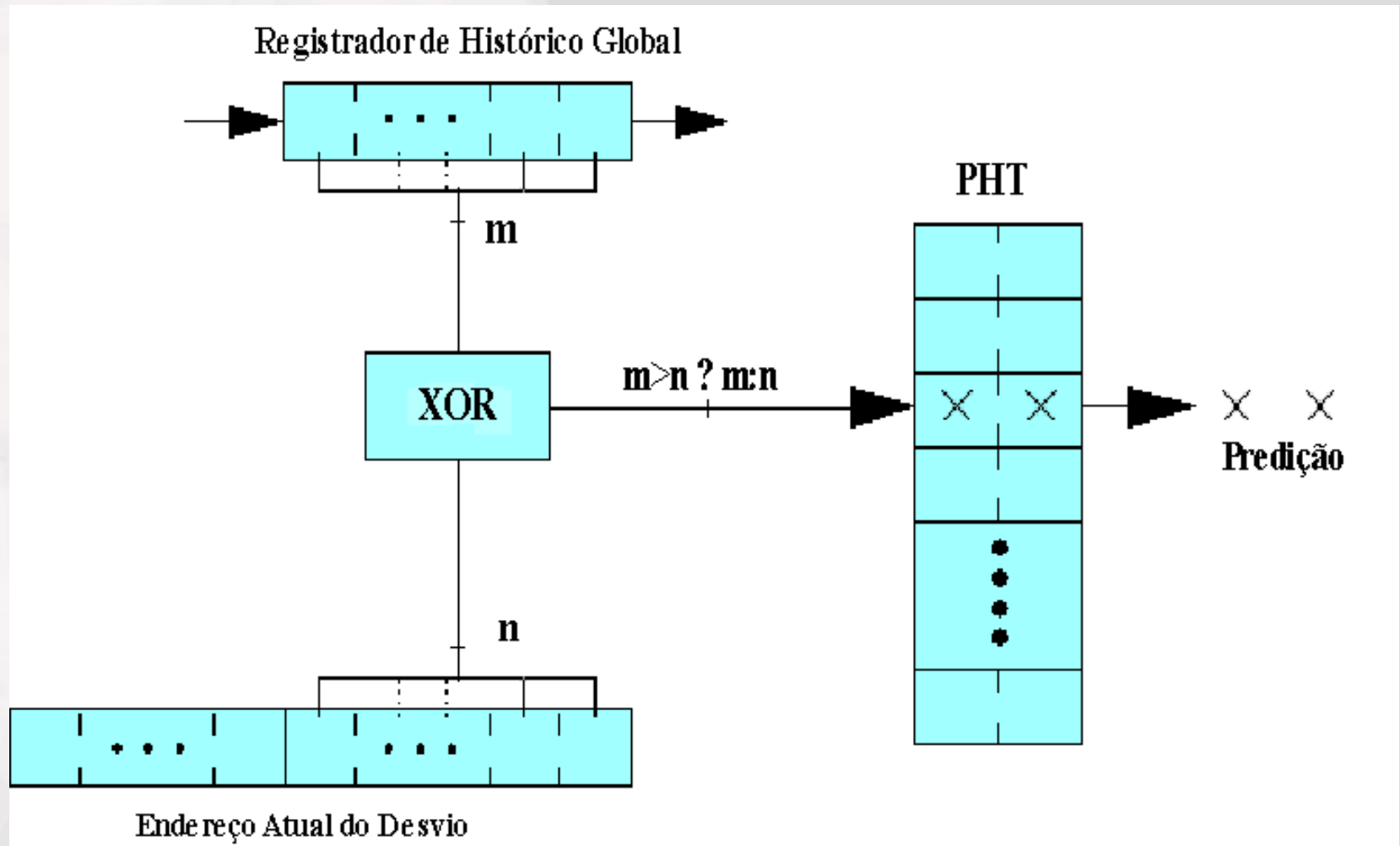
Gselect



Preditor Gshare

- Utiliza o **OU exclusivo** bit a bit de parte do endereço de desvio com os bits do registrador de histórico (BHR) como função HASH.
- Assim, o acesso à PHT também é feito levando em conta o comportamento dos últimos desvios executados.
- A figura a seguir ilustra o funcionamento deste tipo de preditor.

GShare

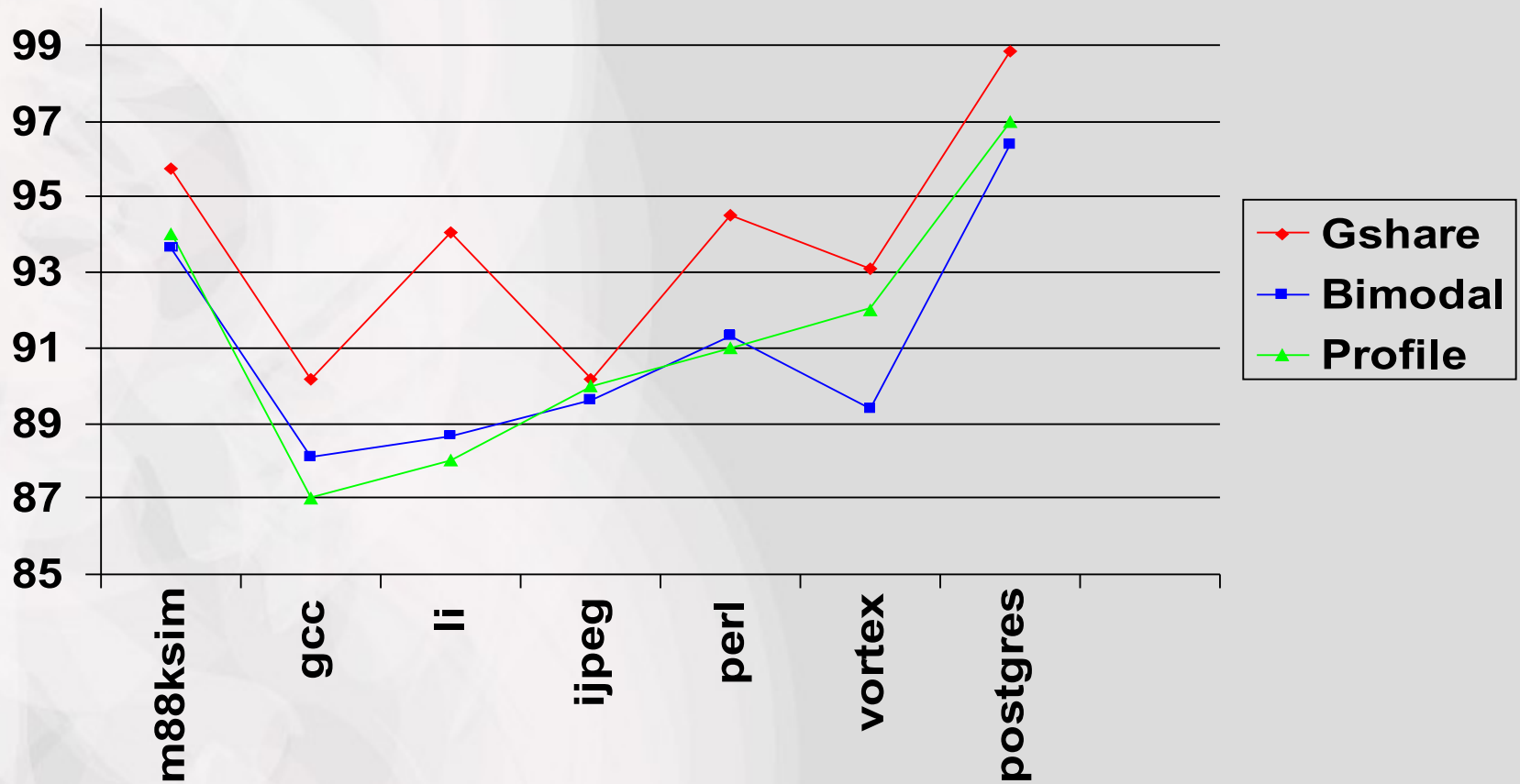


Preditores gselect x gshare

- **Preditor gselect:** concatena alguns bits de mais baixa ordem do endereço do desvio e os bits do registrador de histórico global (BHR)
- **Preditor gshare:** Utiliza o OU exclusivo bit a bit de parte do endereço de desvio com os bits do registrador de histórico como função HASH.
- **McFarling:** gshare é ligeiramente melhor que o gselect.

<u>Branch Address</u>	<u>BHR</u>	<u>gselect4/4</u>	<u>gshare8/8</u>
00000000	00000001	00000001	00000001
00000000	00000000	00000000	00000000
11111111	00000000	11110000	11111111
11111111	10000000	11110000	01111111

Preditor Gshare



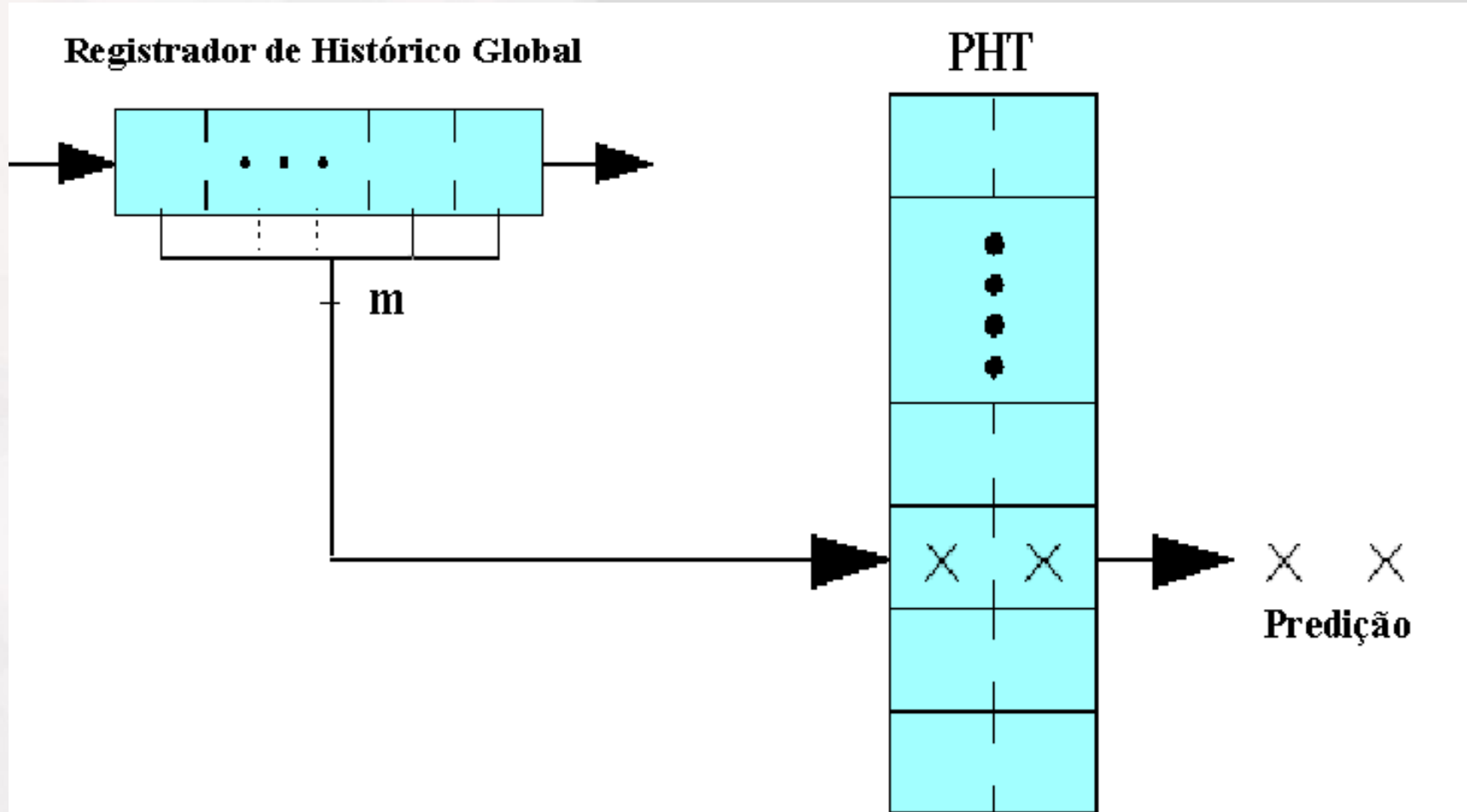
Preditor GAg

- Neste caso o endereçamento da PHT é feito apenas pelos m bits do registrador de histórico global (BHR), ou seja, é baseada apenas no padrão de histórico dos m desvios mais recentes.
- Nenhuma informação dos bits do endereço atual do desvio é utilizada.
- O valor do BHR é utilizado para indexar a PHT e encontrar o contador de dois bits correspondente, que é um preditor bimodal.
- Isto caracteriza uma predição de desvio global, onde se considera que a direção tomada pelo desvio atual depende fortemente do comportamento de outros desvios.

Preditor GAg

- A PHT tem necessariamente 2^m entradas, com cada posição correspondendo a um contador de 2 bits.
- Depois que o desvio condicional for resolvido, o resultado é colocado no BHR, deslocando o bit mais à esquerda para fora do registrador, e o contador de 2 bits é incrementado em um desvio tomado e decrementado em um desvio não tomado.
- Apesar de estranho, este preditor tem um funcionamento melhor que o preditor bimodal de um nível.

Preditor GAg



Predição de Dois Níveis

- **Usa dois níveis de informação para fazer uma predição de direção:**
 - **Branch History Table (BHT)**
 - **Pattern History Table (PHT)**
- **Captura padrões de comportamento dos desvios:**
 - **Grupos de desvios são correlacionados**
 - **Desvios particulares tem comportamento particular**

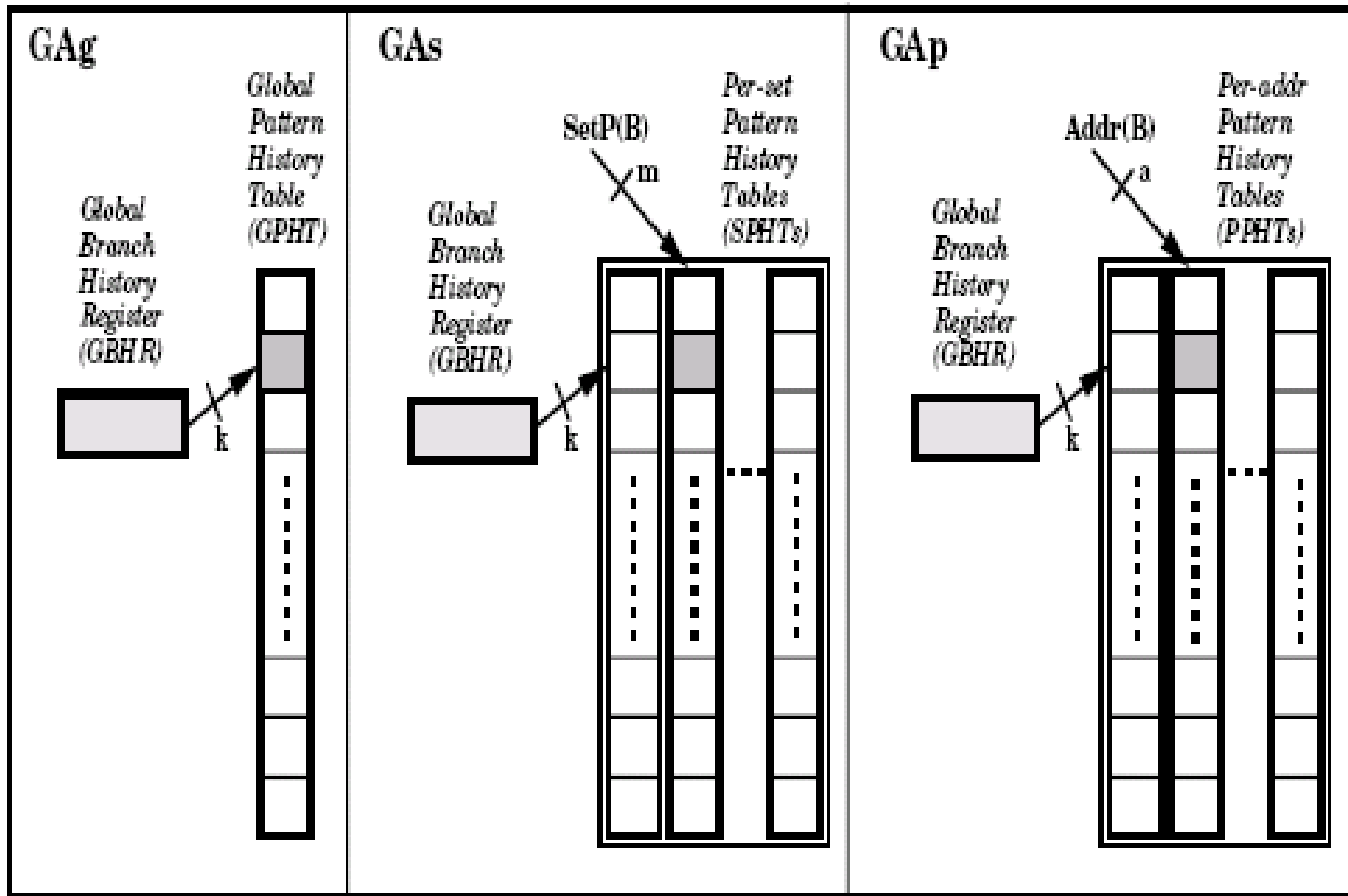
Preditores de Dois Níveis

- Nomenclatura de 3 letras proposta por Yeh e Patt:
- A primeira letra define o tipo de histórico de desvio coletado:
 - **G** (global), **P** (por desvio), **S** (por conjunto)
- A segunda letra o tipo de PHT
 - **A** (adaptativo), **S** (estático)
- A terceira letra define a organização da PHT:
 - **g** (global), **p** (por desvio), **s** (por conjunto)

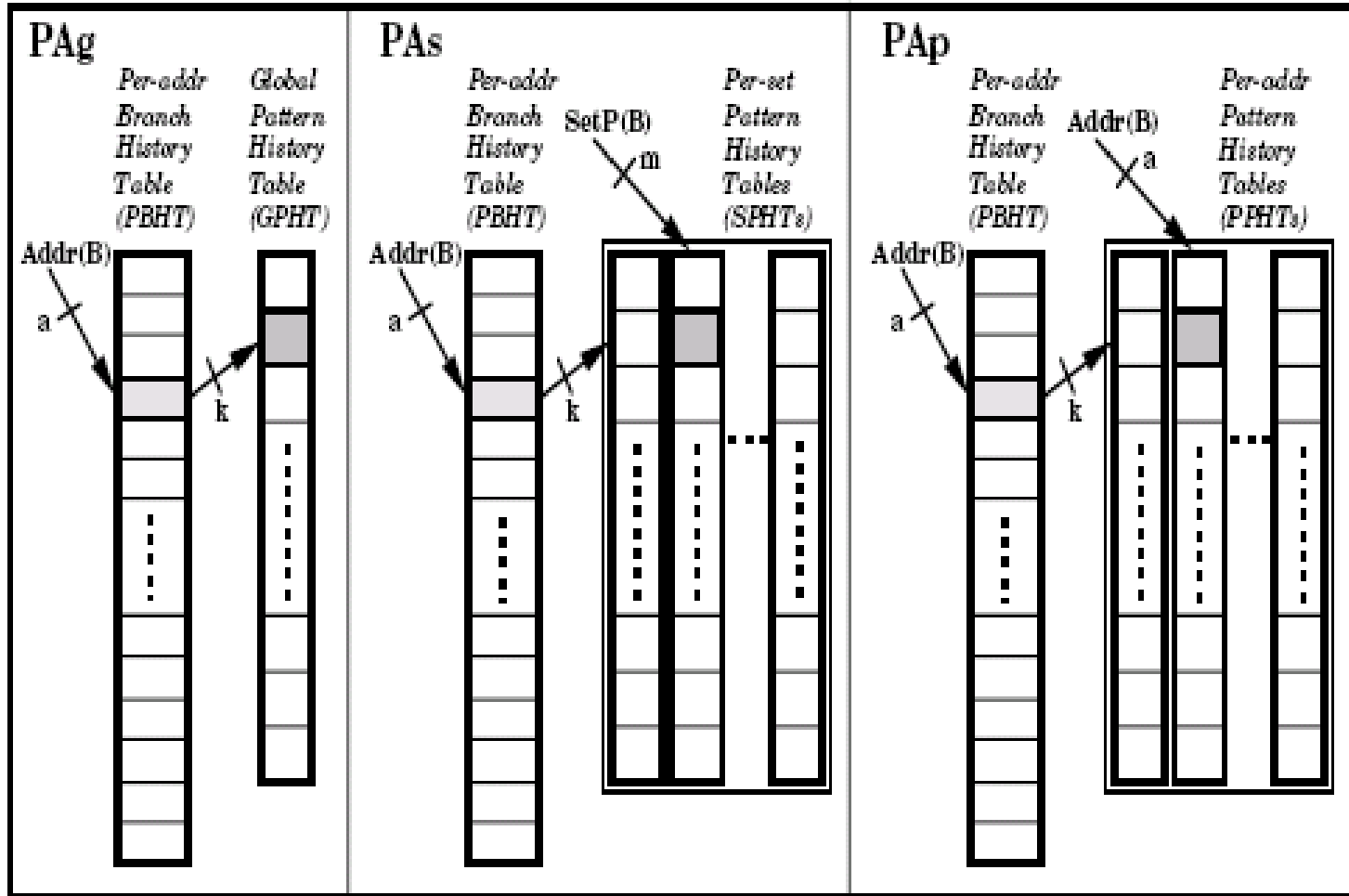
Preditores de Dois Níveis

	PHT Global Única	PHT por Conjunto	PHT por Endereço
BHR Global Único	GAg	GAs	GAp
BHT por Endereço	PAg	PAs	PAp
BHT por Conjunto	SAg	SAs	SAp

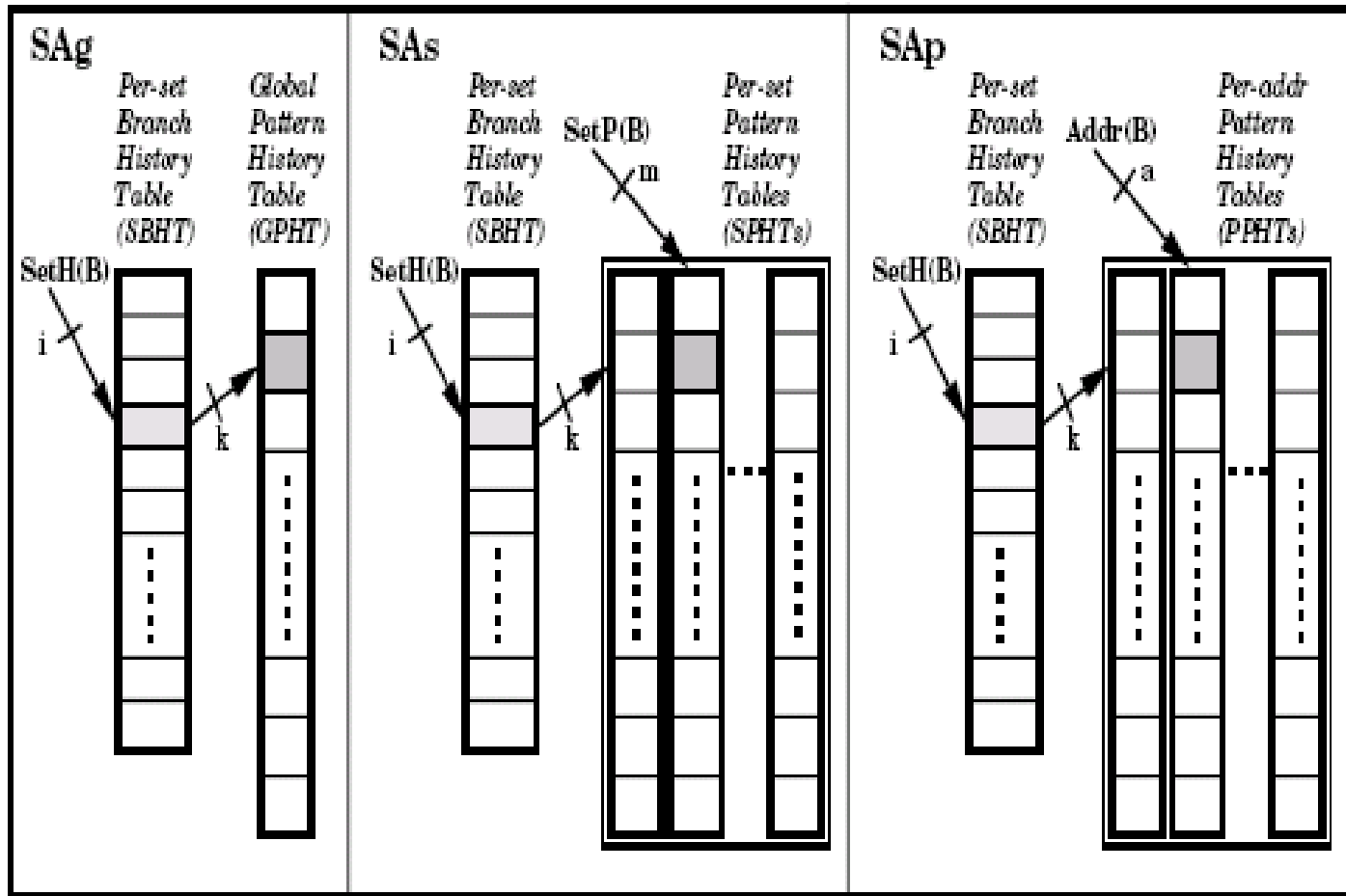
Preditores de Dois Níveis



Preditores de Dois Níveis



Preditores de Dois Níveis



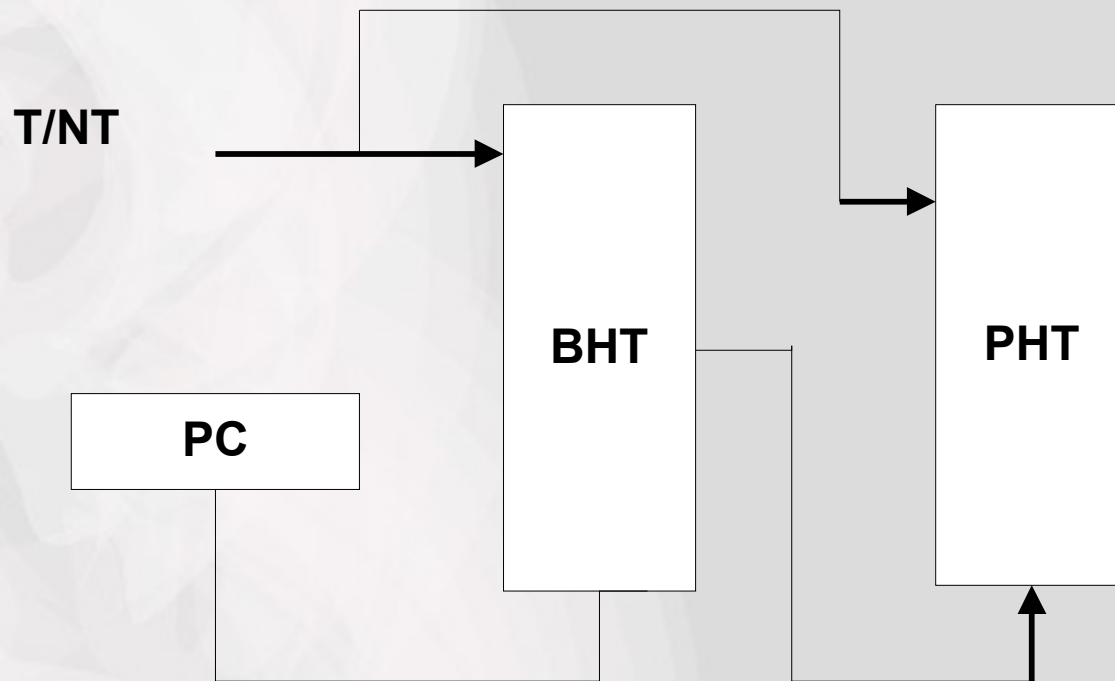
PAg

- **Existe um registrador de histórico de desvio (BHR) para cada desvio, isto é, uma tabela de registradores de histórico de desvio (BHT).**
- **O endereço do desvio condicional é utilizado em uma função de hash para acesso à tabela.**
- **Cada entrada na BHT é um registrador de histórico de desvio com m bits, registrando se o m desvios mais recentes correspondentes a essa entrada foram tomados ou não.**
- **Existe uma outra tabela de padrões de histórico de desvio (PHT), que é a mesma do GAg.**
- **A predição de um desvio é baseada no padrão de histórico dos m resultados de desvios.**

PAg

- **Sempre que um desvio condicional é encontrado, o seu endereço é utilizado para obtermos o seu registrador de histórico de desvios (BHR) na entrada correspondente da BHT .**
- **Esse BHR é então utilizado para endereçar a PHT e obter uma predição.**
- **Depois que o desvio condicional for resolvido, o resultado é colocado à esquerda do BHR, no bit com posição menos significativa e é também utilizado para atualizar o contador de 2 bits na entrada correspondente da PHT.**

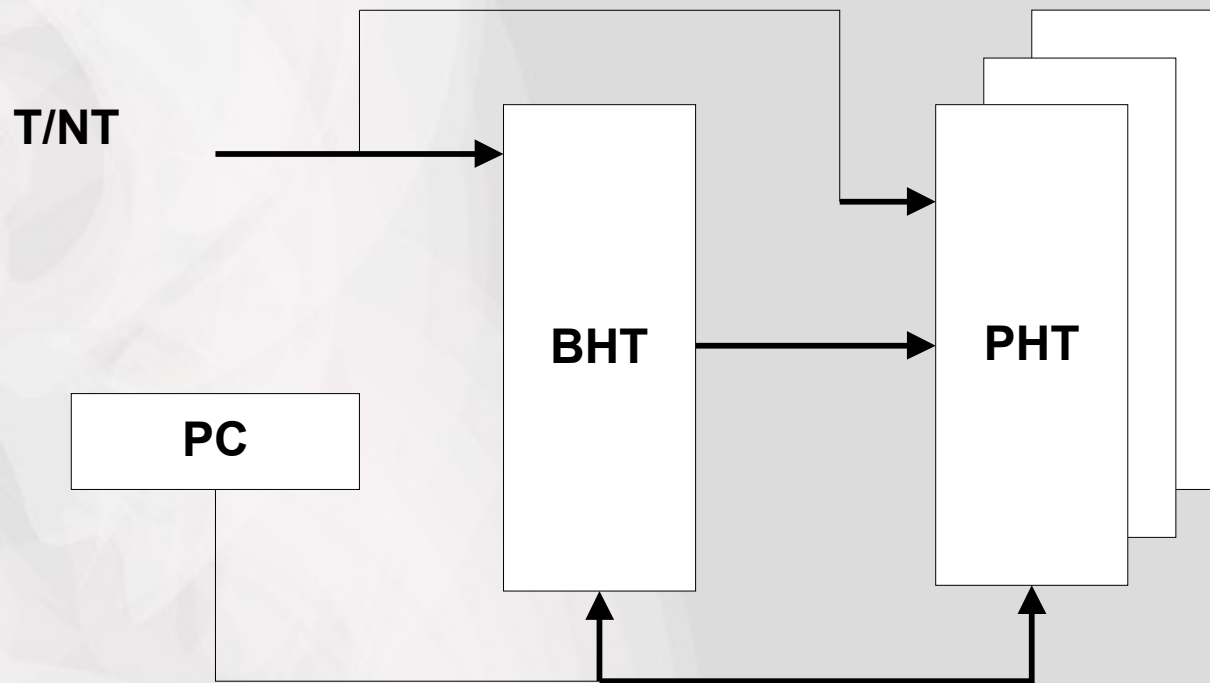
PAg



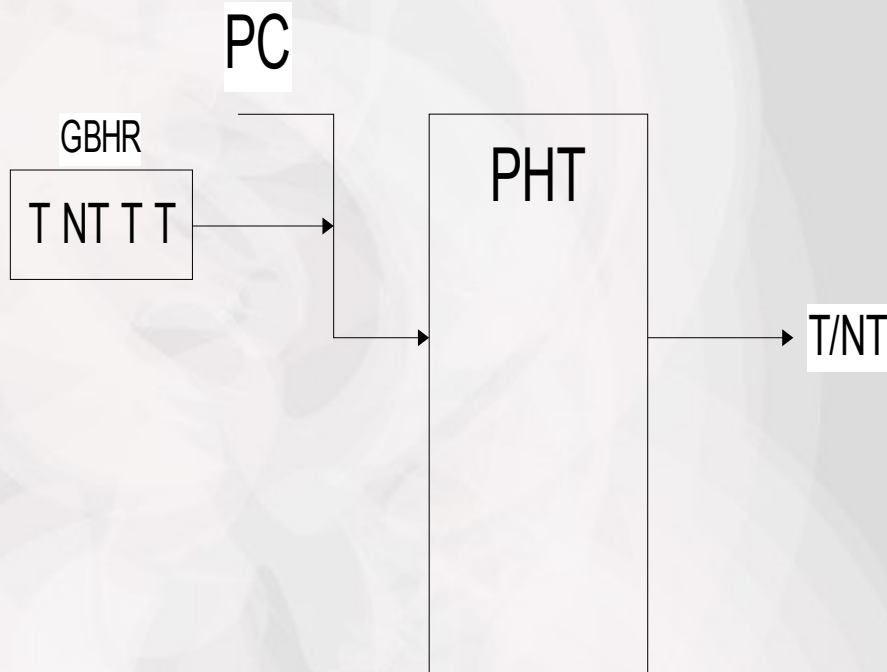
PAp

- **Existe uma PHT para cada desvio, isto é, para cada desvio existe um registrador de histórico de desvio (BHR) e uma tabela de padrões de histórico de desvio (PHT) específica.**
- **Sempre que um desvio condicional é encontrado, localizamos o seu BHR na entrada correspondente na BHT, que é utilizada para indexar a PHT.**
- **A PHT a ser indexada pelo BHR é escolhida dentre todas as PHTs disponíveis pelos bits menos significativos do endereço.**
- **Depois que o desvio condicional for resolvido, o resultado é deslocado à esquerda na posição do bit menos significativo do BHR e utilizado para atualizar o contador de 2 bits na PHT escolhida.**

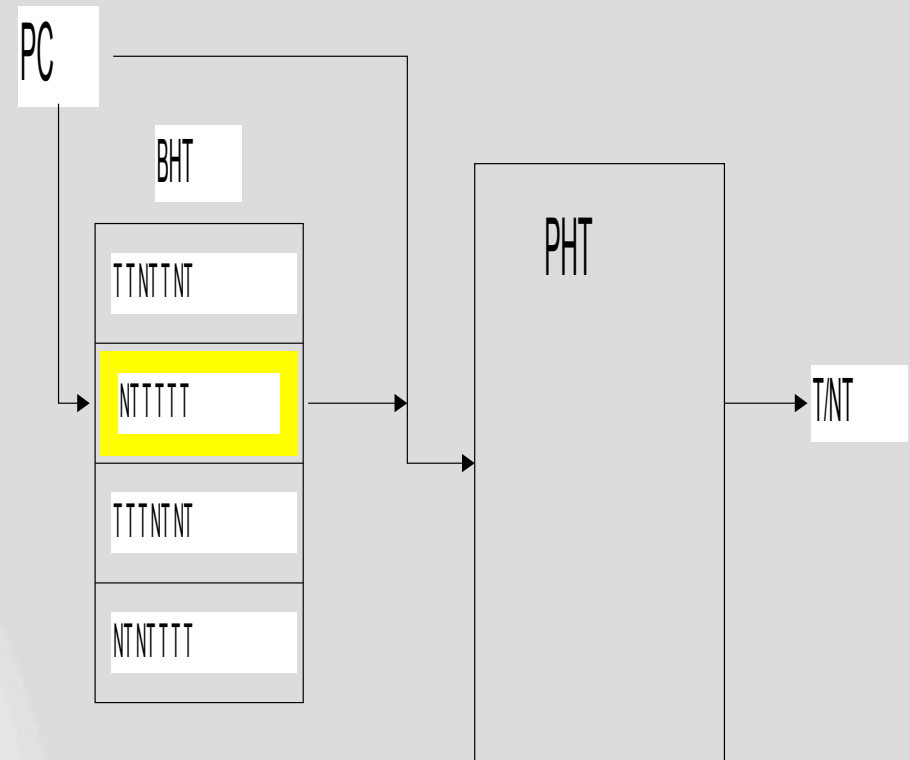
PAp



Preditores de Dois Níveis



Preditor GAs



Preditor PAs

Preditores Híbridos

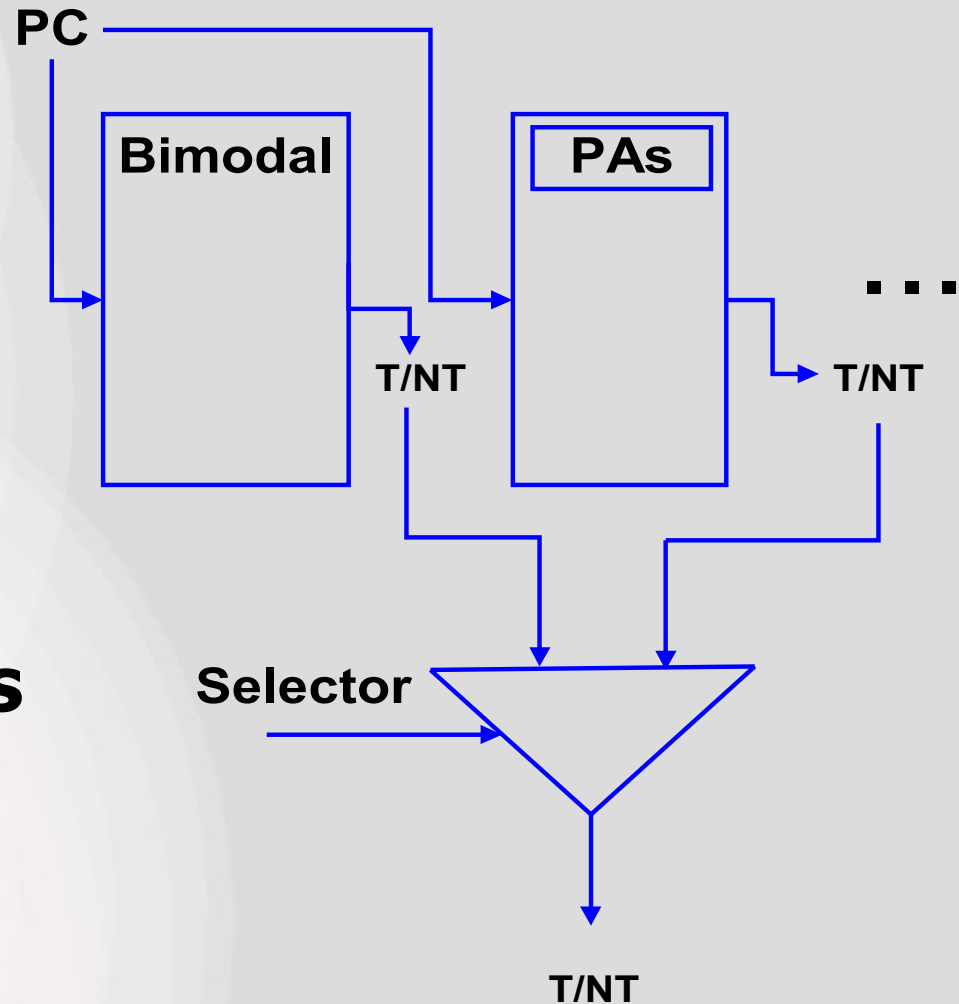
- **Se um preditor local tem desempenho superior a um preditor global para um desvio A , então o desvio A é dito ter um comportamento local.**
- **Os desvios tendem a ter, durante boa parte do tempo, um comportamento local ou global.**
- **Contudo, esses desvios podem mudar de comportamento durante a execução de um programa.**
- **Dado que os diferentes preditores de desvios discutidos tem diferentes vantagens, procurou-se elaborar um novo tipo de preditor de desvio que combinasse as vantagens de todos.**

Preditores Híbridos

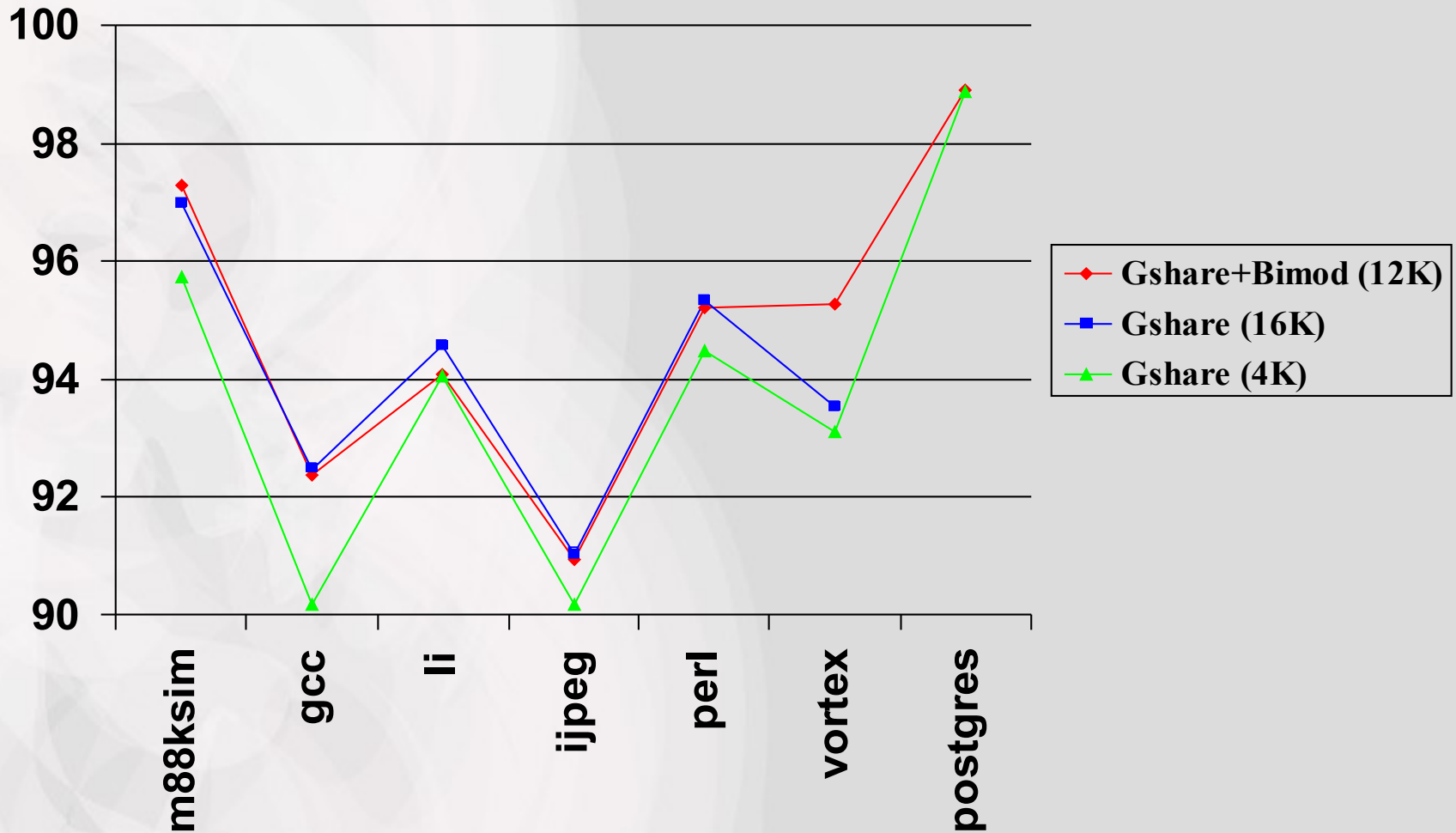
- **Um dos esquemas mais importantes foi proposto por Scott McFarling e combina preditores locais e globais em um preditor híbrido.**
- **O preditor híbrido monitora qual tipo de preditor tem um desempenho melhor para um dado desvio e usa uma variedade de mecanismos de seleção para escolher entre eles.**

Preditores Híbridos

- **Dois ou mais preditores combinados**
- **Desvios distintos se beneficiam de diferentes tipos de histórico**



Preditores Híbridos



Simulações de Grunwald 1998

Aplicação	instruções	desvios	desvios	taxa de falhas		
	completadas (milhões)	condicionais (milhões)	tomados (%)	SAg	gshare	híbrido
compress	80.4	14.4	54.6	10.1	10.1	9.9
gcc	250.9	50.4	49.0	12.8	23.9	12.2
perl	228.2	43.8	52.6	9.2	25.9	11.4
go	548.1	80.3	54.5	25.6	34.4	24.1
m88ksim	416.5	89.8	71.7	4.7	8.6	4.7
xlisp	183.3	41.8	39.5	10.3	10.2	6.8
vortex	180.9	29.1	50.1	2.0	8.3	1.7
jpeg	252.0	20.0	70.0	10.3	12.5	10.4
média	267.6	46.2	54.3	8.6	14.5	8.1

Tabela de Instruções de Desvio

- Uma tabela com as instruções de desvio condicional mais recentes que **não** foram tomados. Se um desvio condicional está na tabela a predição será **não tomado**; do contrário será predito como **tomado**.
- São retirados da tabela os desvios tomados e é utilizado um esquema LRU para adicionar novas entradas.

Fatores que Influenciam a Predição

- **Erros de predição ocorrem quando um desvio está se comportando de uma maneira enquanto que a predição supõe um outro tipo de comportamento.**
- **Preditores locais consideram cada desvio independentemente, enquanto que preditores globais combinam o histórico de todos os desvios recentes para realizar uma predição.**

Fatores que Influenciam a Predição

- **Além do fato de que a maioria dos programas tem alguns desvios que necessitam de tipo de preditor local enquanto outros precisam de um tipo de preditor global, os desvios *individualmente* mudam *com* frequência de comportamento: às vezes necessitam de um histórico local, outras vezes de um histórico global.**
- **Isto diminui as taxas de acerto, mesmo quando se utiliza um preditor híbrido.**

Fatores que Influenciam a Predição

- **Aliasing**
 - Mais de um desvio pode utilizar a mesma entrada na BHT/PHT
 - **Construtiva**
 - Predição que poderia ser incorreta, predita corretamente
 - **Destrutiva**
 - Predição que poderia ser correta, predita incorretamente
 - **Neutra**
 - Nenhuma mudança na acurácia

Fatores que Influenciam a Predição

- **Tempo de treinamento**
 - É necessário haver desvios suficientes para um padrão ser descoberto
 - É necessário algum tempo até se entrar em um estado estável
- **Histórico “Errado”**
 - Tipo incorreto de histórico para o desvio
- **Troca de contexto**
 - “Aliasing” causado por desvios de diferentes programas

Desvios Especiais

- **Chamadas e Retorno de Procedimento**
 - Chamadas são sempre tomadas
 - O endereço de retorno quase sempre é conhecido
- **Return Address Stack (RAS)**
 - Na chamada de um procedimento, colocar o endereço da instrução seguinte na pilha de endereço de retorno

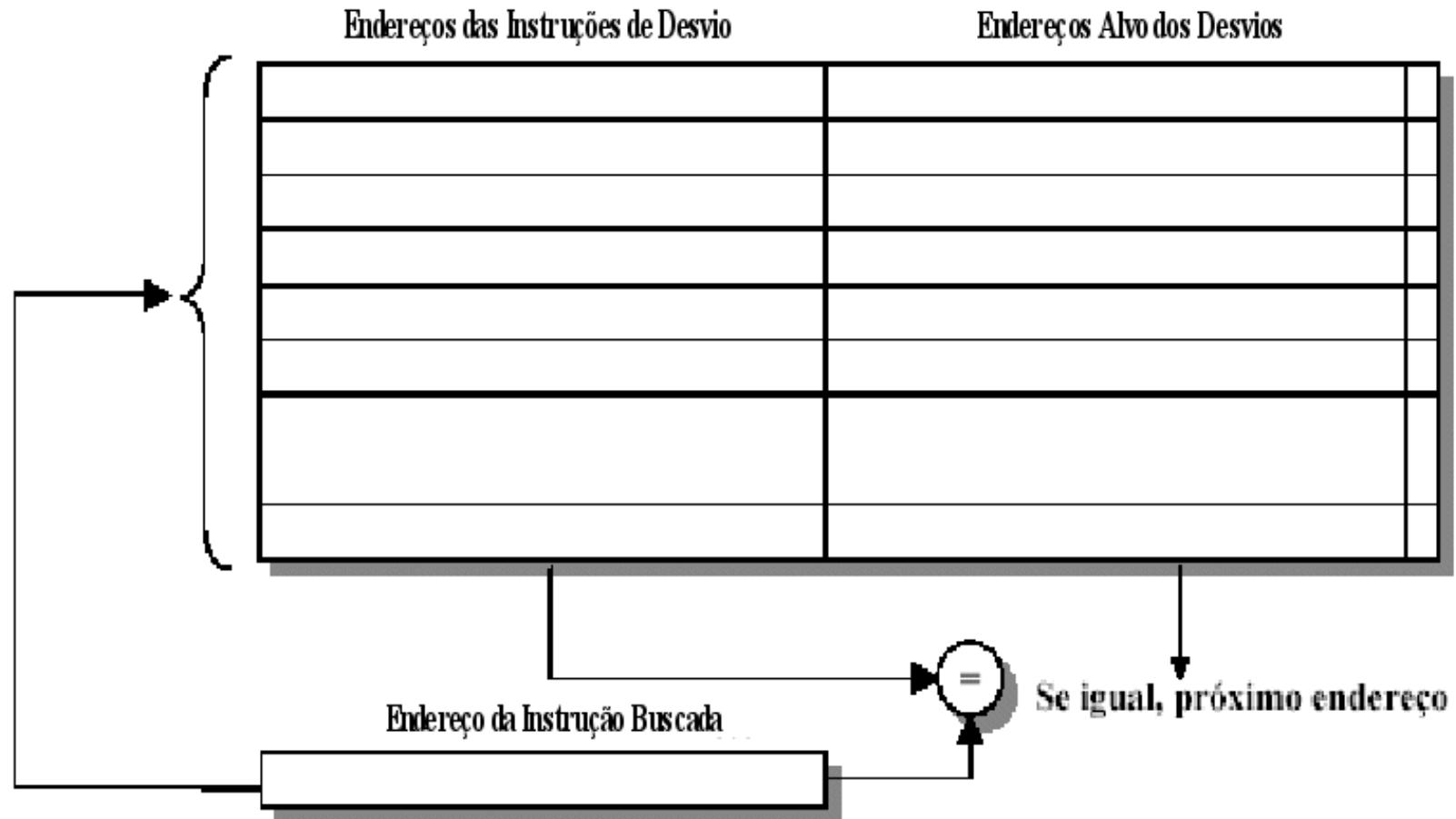
Branch Target Address Cache

- **É uma cache especial, que armazena o endereço da instrução de desvio e o endereço alvo dos últimos desvios tomados.**
- **É consultada durante o estágio de busca das instruções. Se houver acerto, indica que a instrução que está sendo buscada é um desvio, e o endereço alvo já pode ser obtido diretamente da cache.**
- **Se a predição indicar que o desvio será tomado, esse endereço é utilizado para a busca das próximas instruções.**

Branch Target Address Cache

- **Se houver uma falha na BTAC, a instrução é considerada como uma instrução normal ou uma instrução de desvio com a predição de não ser tomada. Então:**
 - **Se a instrução não causar realmente um desvio, o pipeline continua operando normalmente**
 - **Se entretanto a instrução provar ser uma instrução de desvio tomado, a BTAC também é atualizado com o valor do PC da instrução alvo do desvio**

Branch Target Address Cache



Branch Target Address Cache

- **Desempenho:**

- **Hipótese:** Em caso de falha no BTAC ou na predição, o desvio gasta 2 ciclos, em caso contrário, apenas 1 ciclo.
- **Taxa de acerto no BTAC: 90%**
- **Acurácia: 90%**
- **Frequência de desvios tomados: 60%**
- **Número de ciclos médio por instrução de desvio:**

$$N = 0,9 \times 0,9 \times 1 + 0,1 \times 0,4 \times 1 + 0,9 \times 0,1 \times 2 + 0,1 \times 0,6 \times 2 \\ = 1,15$$

$$N' = 0,6 \times 2 + 0,4 \times 1 = 1,6$$

Branch Target Instruction Cache

- É uma cache similar à BTAC, só que ao invés de armazenar o endereço alvo, armazena a própria instrução que está no endereço alvo do desvio.
- Com este esquema, é possível a execução do desvio com custo "zero".
- Variações neste esquema incluem armazenar mais de uma instrução do endereço alvo.

Alpha 21264

- Pipeline com 8 estágios e penalidade de falha com 7 ciclos
- Cache de 64 KB, associatividade 2-way com linhas contendo bits de predição
 - Cada bloco com 4 instruções contém a predição para o próximo bloco
- Preditor híbrido
 - 12-bit GAg (PHT com 4K entradas com contadores de 2 bits)
 - 10-bit PAg (BHT com 1K entradas, PHT com 1K entradas e contadores com 3 bits)
- Tamanho total: $4K*2 + 4K*2 + 1K*10 + 1K*3 = 29K$ bits! (~180,000 transistors)

UltraSPARC-III

- **Pipeline com 14 estágios, predição utilizada nos estágios 2 e 3.**
- **Preditor Gshare com 16K entradas com contadores de 2 bits**
 - **O preditor bimodal faz o XOR dos bits do PC com o registrador de histórico global para reduzir o "aliasing"**
- **Fila de Falhas**
 - **Diminui a penalidade por falha fornecendo as instruções para uso imediato**

Pentium III

- **Predição Dinâmica de Desvio**
 - **BTB com 512 entradas prediz a direção e o endereço alvo**
 - **Histórico de 4-bits usado com o PC para tirar a direção**
- **Preditor de desvio estático para as falhas no BTB**
- **Return Address Stack (RAS) com 4/8 entradas**
- **Penalidades:**
 - **Não Tomado: sem penalidades**
 - **Tomado predito corretamente: 1 ciclo**
 - **Erro de predição: entre 9 e 26 ciclos, com média entre 10-15 ciclos**

AMD Athlon K7

- **Pipeline inteiro de 10 estágios, de ponto flutuante com 15 estágios, o preditor é utilizado no estágio de busca**
- **PHT com 2K entradas bimodal, BTAC com 2K entradas**
- **RAS com 12 entradas**
- **Penalidades:**
 - **Desvio Tomado Predito Corretamente: 1 ciclo**
 - **Desvio Predito Incorretamente: ≥ 10 ciclos**

Bibliografia

- **Static branch prediction**
 - J.E.Smith. **A study of branch prediction strategies.** ISCA-8, 1981.
 - F.Mueller and D.B.Whalley. **Avoiding unconditional jumps by code replication.** PLDI, 1992.
 - T.Ball and J.R.Larus. **Branch prediction for free.** PLDI, 1993.
 - C.Young and M.D.Smith. **Improving the accuracy of static branch prediction using branch correlation.** ASPLOS-6, 1994.
 - F.Mueller and D.B.Whalley. **Avoiding conditional branches by code replication.** PLDI, 1995.

Bibliografia

- **Semi-static branch prediction**

- S.McFarling and J.Hennessy. **Reducing the cost of branches.** ISCA-13, 1986.
- D.E.Wall. **Predicting program behavior using real or estimated profiles.** PLDI, 1991.
- J.A.Fisher and S.M.Freudenberger. **Predicting conditional branch directions from previous runs of a program.** ASPLOS-5, 1992.
- A.Krall. **Improving semi-static branch prediction by code replication.** PLDI, 1994.
- B.Calder and D.Grunwald. **Reducing branch costs via branch alignment.** ASPLOS-6, 1994.
- B.Calder, D.Grunwald, D.Lindsay, J.Martin, M.Mozer and B.Zorn. **Corpus-based static branch prediction.** PLDI, 1995.

Bibliografia

- **Bimodal predictors and BTBs**
 - J.E.Smith. **A study of branch prediction strategies.** ISCA-8, 1981.
 - J.Lee and A.Smith. **Branch prediction strategies and branch target buffer design.** IEEE Computer 21(7). 1984.
 - S.McFarling and J.Hennessy. **Reducing the cost of branches.** ISCA-13, 1986.
 - T.Yeh and Y.N.Patt. **A comprehensive instruction fetch mechanism for a processor supporting speculative execution.** MICRO-25, 1992.
 - B.Calder and D.Grunwald. **Fast & accurate instruction fetch and branch prediction.** ISCA-21, 1994.

Bibliografia

- **Two-level branch predictors**
 - S.Pan, K.So and J.Rahmeh. **Improving the accuracy of dynamic branch prediction using branch correlation.** ASPLOS-5, 1992.
 - T.Yeh and Y.N.Patt. **Alternative implementations of two-level adaptive branch prediction.** ISCA-19, 1992.
 - T.Yeh and Y.N.Patt. **A comparison of dynamic branch predictors that use two levels of branch history.** ISCA-20, 1993.
 - S.McFarling. **Combining branch predictors.** Technical note TN-36, DEC-WRL, 1993.

Bibliografia

- **Combining branch predictors**

- S.McFarling. **Combining branch predictors**. Technical note TN-36, DEC-WRL, 1993.
- P.Y.Chang, E.Hao and Y.N.Patt. **Alternative Implementations of hybrid branch predictors**. MICRO-28, 1995.

- **Dynamic history length**

- T.Juan, S.Sanjeevan and J.Navarro. **Dynamic history length fitting: a third level of adaptivity for branch prediction**. ISCA-25, 1998.
- J.Stark, M.Evers and Y.N.Patt. **Variable length path branch prediction**. ASPLOS-8, 1998.

Bibliografia

- **Novel branch predictors**

- C.C.Lee, I.Chen and T.Mudge. **The bi-mode branch predictor.** MICRO-30 1997.
- E.Sprangle, R.Chappell, M.Alsup and Y.N.Patt. **The agree predictor: a mechanism for reducing negative branch history interference.** ISCA-24, 1997.
- J.Gonzalez and A.Gonzalez. **Control-flow speculation through value prediction for superscalar processors.** PACT 1999.

- **Trace predictors**

- Q.Jacobson, E.Rottenberg and J.E.Smith. **Path-based next trace prediction.** MICRO-30, 1997.
- B.Black, B.Ryckick and J.P.Shen. **The block-based trace cache.** ISCA-26, 1999.

Bibliografia

- **Multiple branch predictors**
 - T.Yeh, D.Marr and Y.N.Patt. **Increasing instruction fetch rate via multiple branch prediction and a branch address cache.** SC-7, 1993.
 - S.Dutta and M.Franklin. **Control flow prediction with tree-like subgraphs for superscalar processors.** MICRO-28, 1995.
 - T.Conte, K.Menezes, P.Mills and B.Patel. **Optimization of instruction fetch mechanisms for high issue rates.** ISCA-22, 1995.
 - S.Wallance and N.Bagherzadeh. **Multiple branch and block prediction.** 1997.