2024年度（令和6年）版

Ver. 2024-11-01a

Course number: CSC.T363

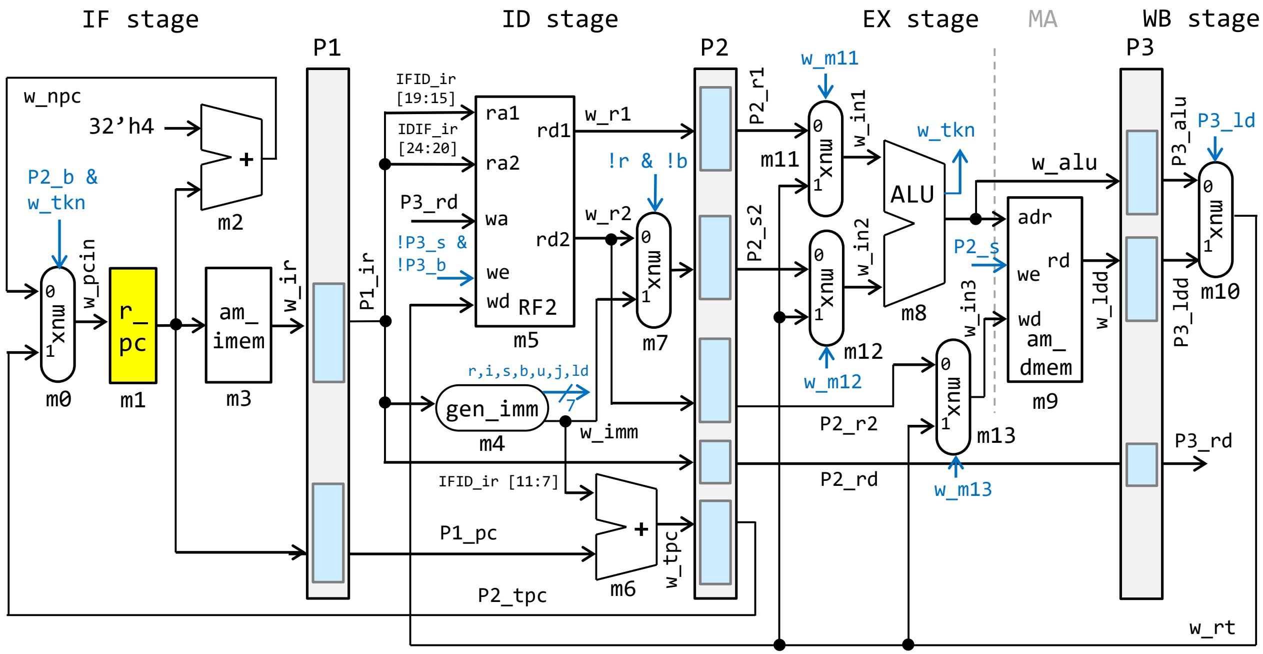
コンピュータアーキテクチャ
Computer Architecture

9. 分岐予測 (2)
Branch Prediction (2)

www.arch.cs.titech.ac.jp/lecture/CA/
Tue 13:30-15:10, 15:25-17:05
Fri 13:30-15:10

吉瀬 謙二 情報工学系
Kenji Kise, Department of Computer Science
kise _at_ c.titech.ac.jp

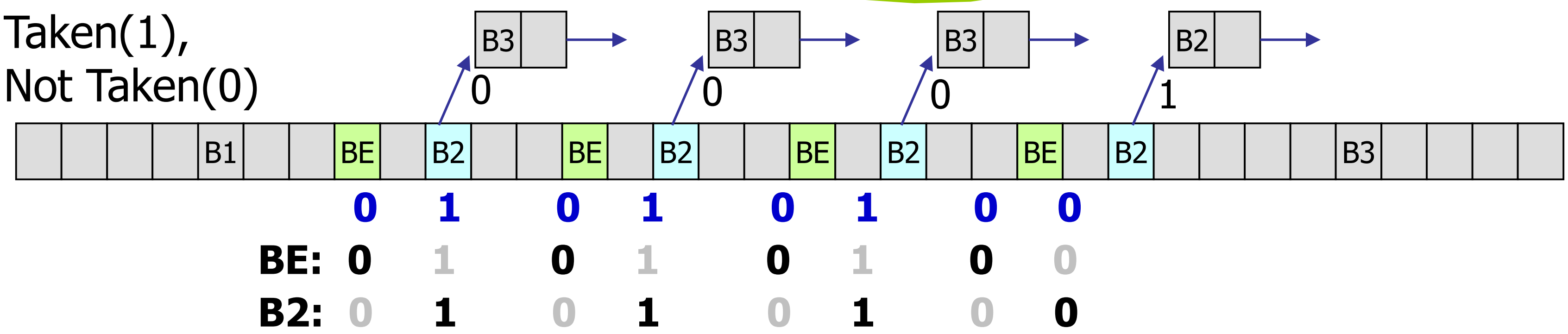
proc8: 4-stage pipelining processor



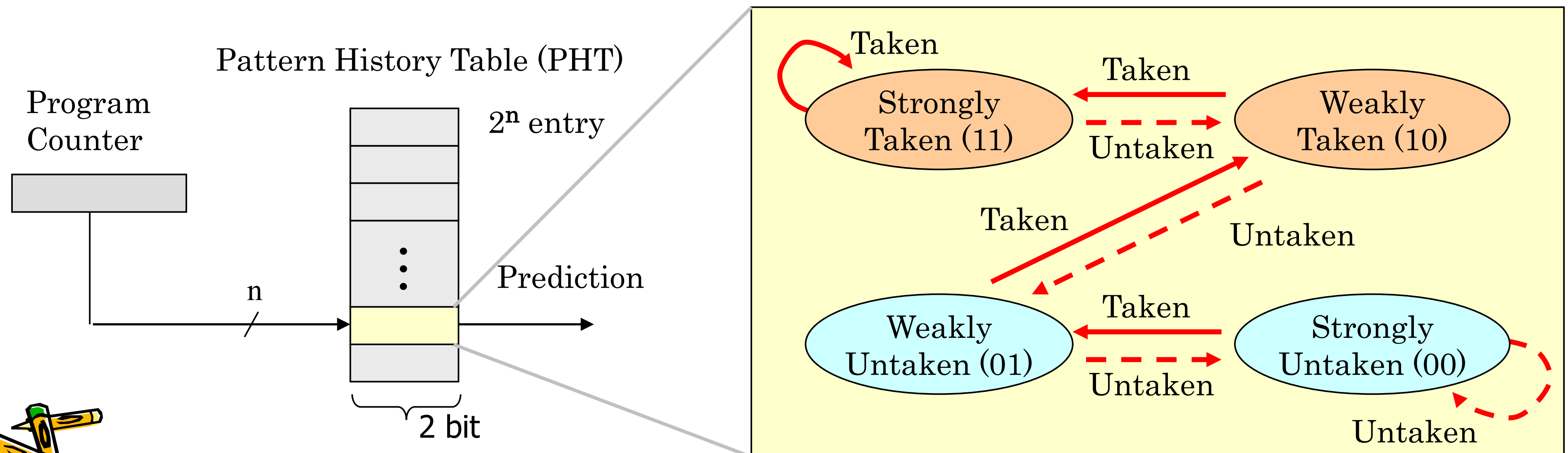
Instructions		CC1	CC2	CC3	CC4	CC5	CC6	CC7	Time
32'h00	addi x1,x0,3	IF	ID	EX	**WB**				
32'h04	bne x2,x1,L		IF	ID	**EX**	WB			
32'h08	addi x10,x1,5			IF	ID	EX	WB		
32'h0c	addi x10,x10,1				IF	ID	EX	WB	
32'h10	L:addi x30,x10,0					IF	ID	EX	WB

Bimodal branch predictor (ISCA 1981)

Taken(1),
Not Taken(0)

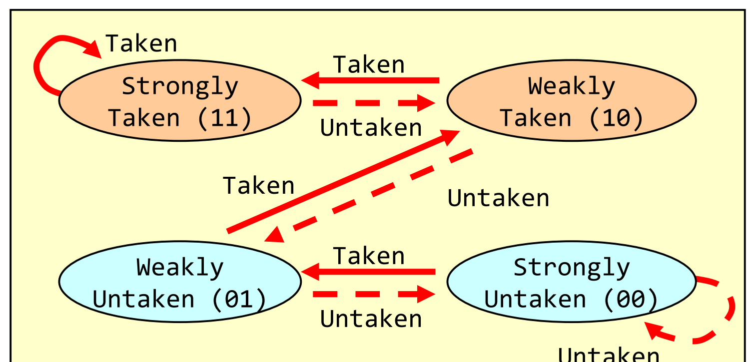


- 分岐アドレス（プログラムカウンタ）毎に履歴を切り替える
  - 分岐アドレスによりパターン履歴表（PHT）のインデックスを作成
- パターン履歴表は2ビットカウンタの配列.



Bimodal branch predictor (ISCA 1981)

```
module m_bimodal (
  input  wire w_clk, w_we,
  input  wire [4:0] w_wadr, w_radr,
  input  wire w_tkn,
  output wire w_pred
);
  reg [1:0] mem [0:31];
  wire [1:0] w_data = mem[w_radr];
  assign w_pred = w_data[1];
  wire [1:0] w_cnt  = mem[w_wadr];
  always @(posedge w_clk) if (w_we)
    mem[w_wadr] <= (w_cnt < 3 &  w_tkn) ? w_cnt + 1 :
                   (w_cnt > 0 & !w_tkn) ? w_cnt - 1 : w_cnt;
  integer i; initial for (i=0; i<32; i=i+1) mem[i] = 1;
endmodule
```



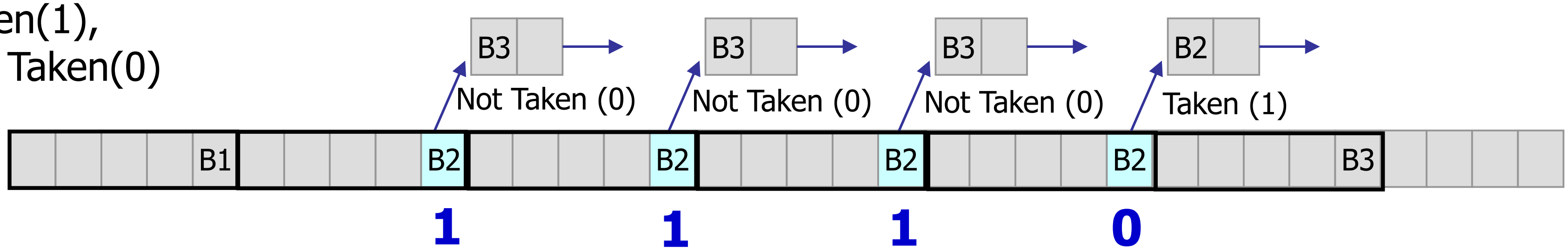
サンプルプログラム

```
`MM[0]={12'd0,5'd0,3'h0,5'd10,7'h13};            // 00   addi x10,x0,0
`MM[1]={12'd0,5'd0,3'h0,5'd3,7'h13};             // 04   addi x3,x0,0
`MM[2]={12'd101,5'd0,3'h0,5'd1,7'h13};           // 08   addi x1,x0,101
`MM[3]={7'd0,5'd3,5'd10,3'h0,5'd10,7'h33};       // 0c L:add  x10,x10,x3
`MM[4]={12'd1,5'd3,3'h0,5'd3,7'h13};             // 10   addi x3,x3,1
`MM[5]={~12'd0,5'd1,5'd3,3'h1,5'b11001,7'h63};// 14   bne  x3,x1,L
`MM[6]=32'h00050f13;                             // 18   HALT
```

```
`MM[0] ={12'd4,  5'd0, 3'h0,5'd1, 7'h13};        // 00   addi x1,x0,4
`MM[1] ={12'd101,5'd0, 3'h0,5'd2, 7'h13};        // 04   addi x2,x0,101
`MM[2] ={12'd0,  5'd0, 3'h0,5'd10,7'h13};        // 08   addi x10,x0,0
`MM[3] ={12'd0,  5'd0, 3'h0,5'd4 ,7'h13};        // 0c   addi x4,x0,0
`MM[4] ={12'd0,  5'd0, 3'h0,5'd3 ,7'h13};        // 10 M:addi x3,x0,0
`MM[5] ={12'd1,  5'd3 ,3'h0,5'd3, 7'h13};        // 14 L:addi x3,x3,1
`MM[6] ={12'd1,  5'd10,3'h0,5'd10,7'h13};        // 18   addi x10,x10,1
`MM[7] ={~12'd0,5'd1,5'd3,3'h1,5'b11001,7'h63};// 1c   bne  x3,x1,L
`MM[8] ={12'd1,  5'd4 ,3'h0,5'd4, 7'h13};        // 20   addi x4,x4,1
`MM[9] ={12'd1,  5'd10,3'h0,5'd10,7'h13};        // 24   addi x10,x10,1
`MM[10]={~12'd0,5'd2,5'd4,3'h1,5'b01001,7'h63};// 28   bne  x4,x2,M
`MM[11]=32'h00050f13;                            // 2c   HALT
```

Branch history（分岐履歴）

Taken(1),
Not Taken(0)



B2の分岐履歴

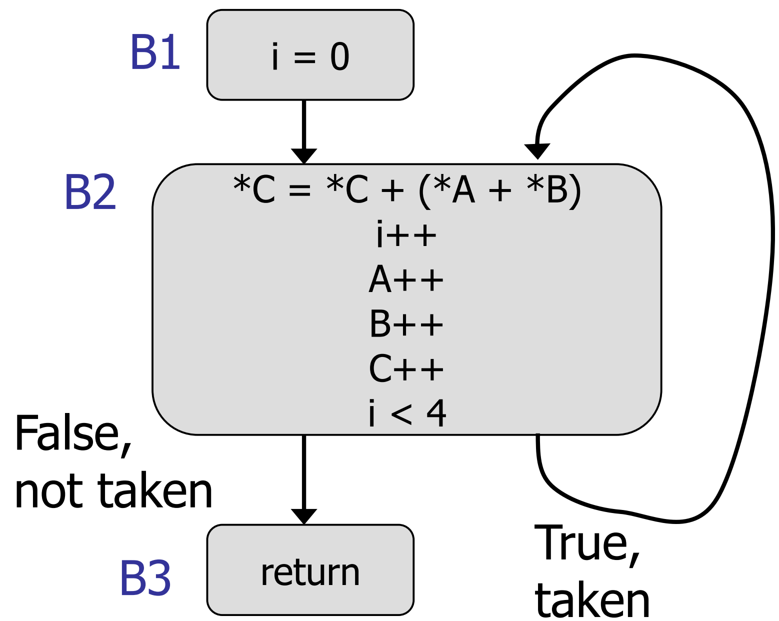
1110 ?

11101 ?

111011 ?

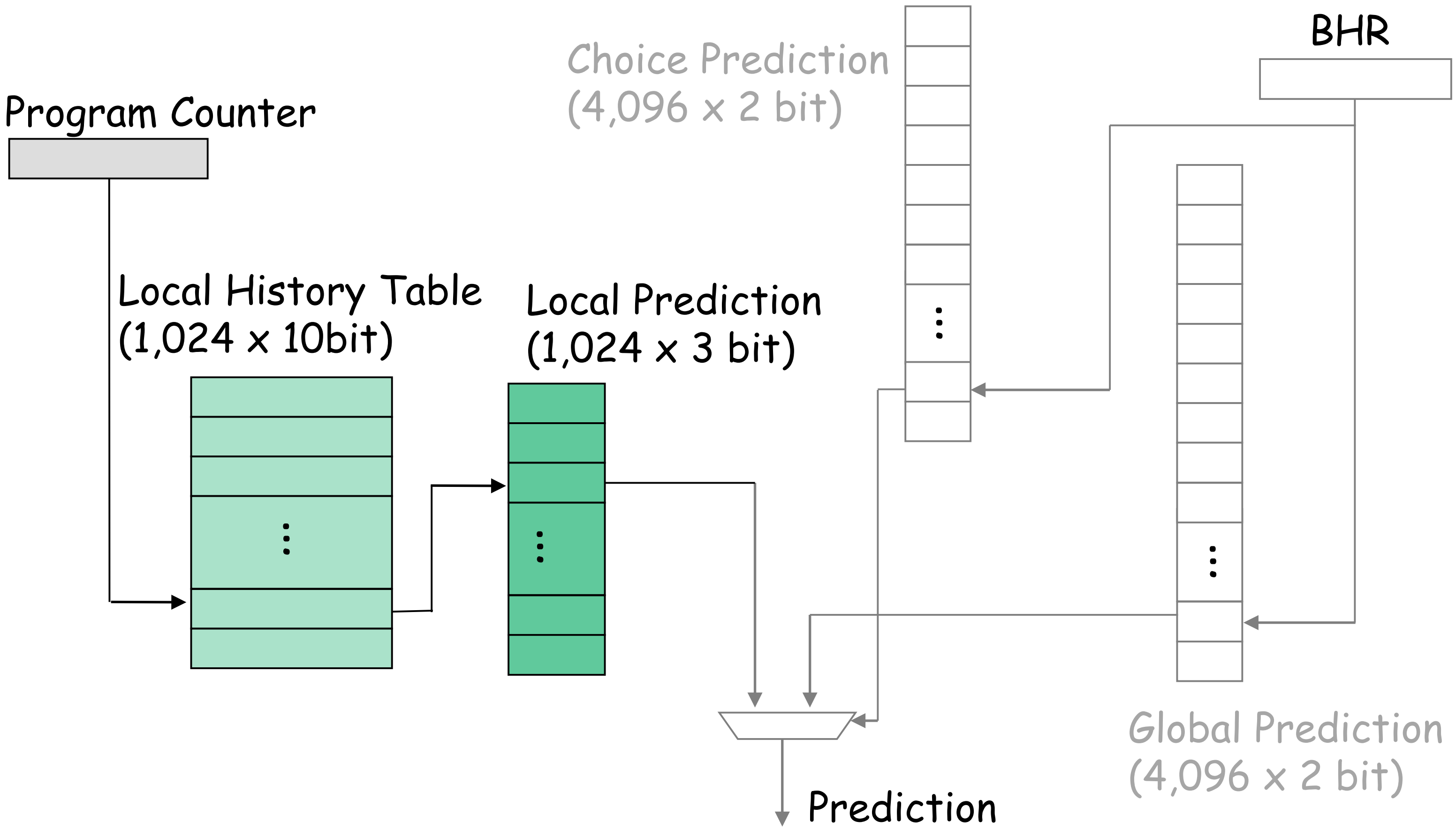
1110111 ?

11101110 ?



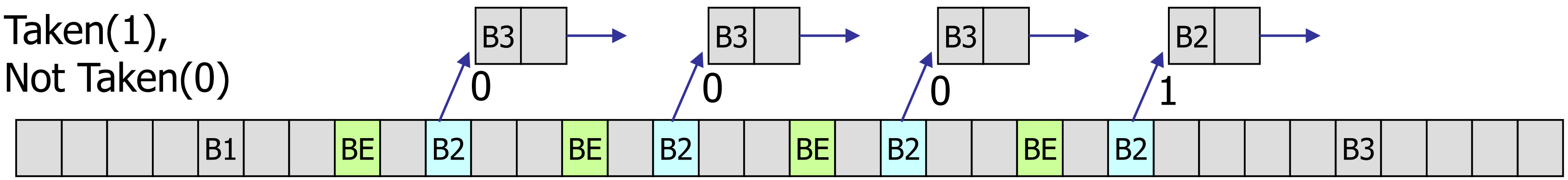
Alpha 21264's hybrid branch predictor in 1996

- A hybrid of local prediction and global prediction implemented in DEC Alpha 21264 which was the state-of-the art commercial processor.
- A choice predictor is used as a meta-predictor



ローカル分岐履歴とグローバル分岐履歴

Taken(1),
Not Taken(0)



	0	**1**	**0**	**1**	**0**	**1**	**0**	**0**
BE:	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**0**
B2:	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**0**

BEの分岐履歴

0000 ?

00000 ?

000000 ?

0000000 ?

00000000 ?

ローカル分岐履歴

B2の分岐履歴

1110 ?

11101 ?

111011 ?

1110111 ?

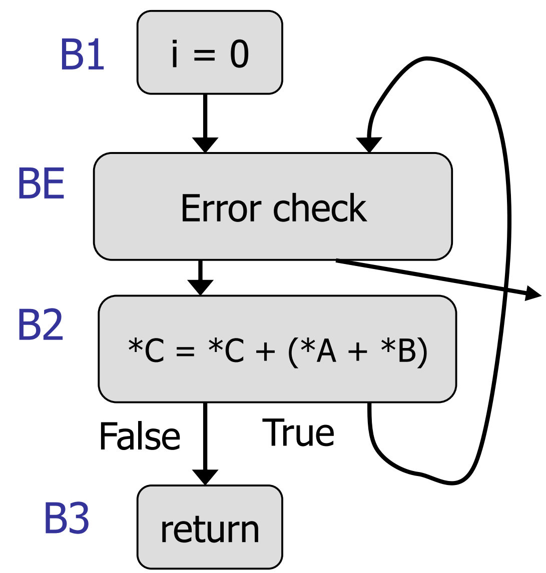
11101110 ?

ローカル分岐履歴

B2とBEの分岐履歴

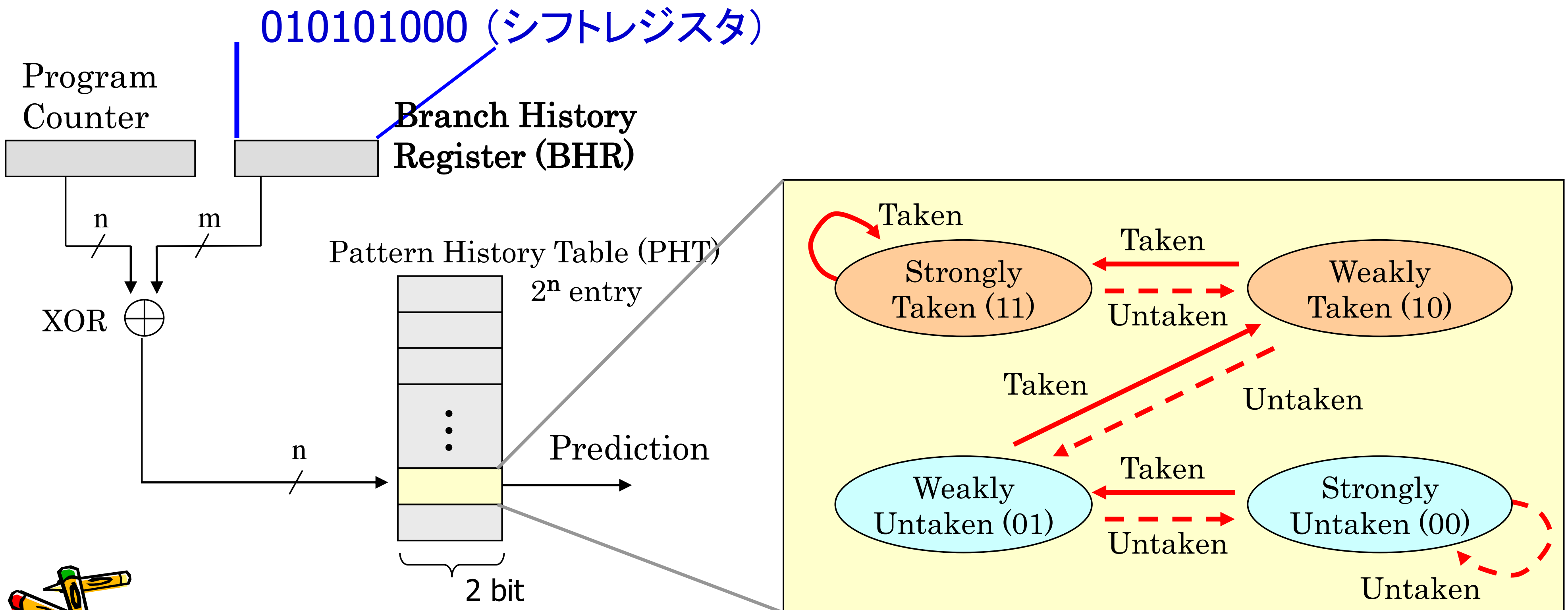
010101000 ?

グローバル分岐履歴



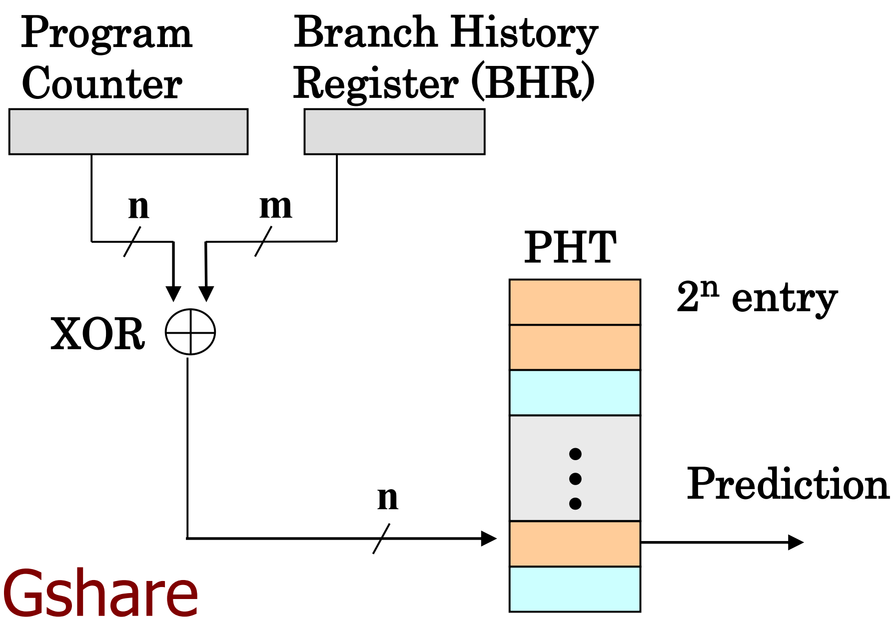
Gshare (TR-DEC 1993)

- グローバル分岐履歴と分岐アドレスとの排他的論理和によりパターン履歴表へのインデックスを作成
  - パターン履歴表は2ビット飽和型カウンタの配列で，選択された2ビットカウンタの値により分岐方向を予測（bimodalと同じ）
  - 分岐結果を用いて，予測に利用したカウンタを更新

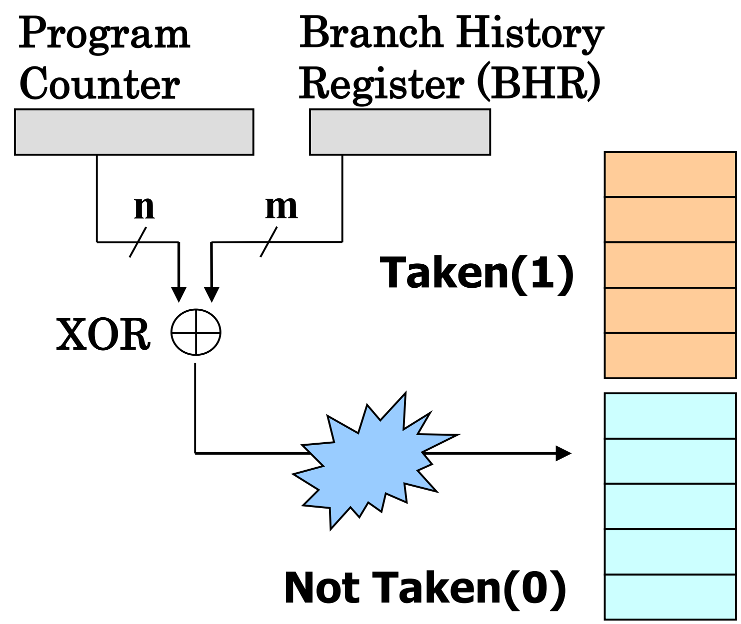


Gshare の改良

- PHTの競合が発生して性能が低下
- PCとBHRによって特定される予測（成立，不成立）には偏りが存在するので，これらを別のテーブルに格納することで競合の悪影響を緩和
  - 分岐成立に偏っているもの
  - 分岐不成立に偏っているもの

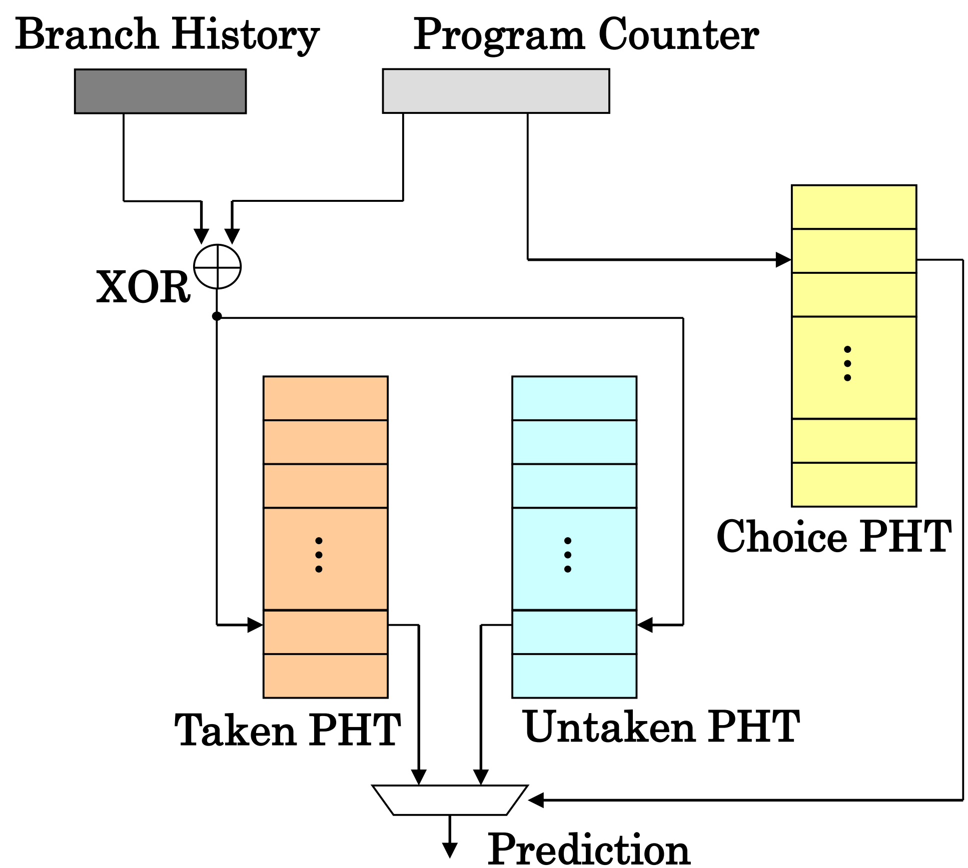


Gshare

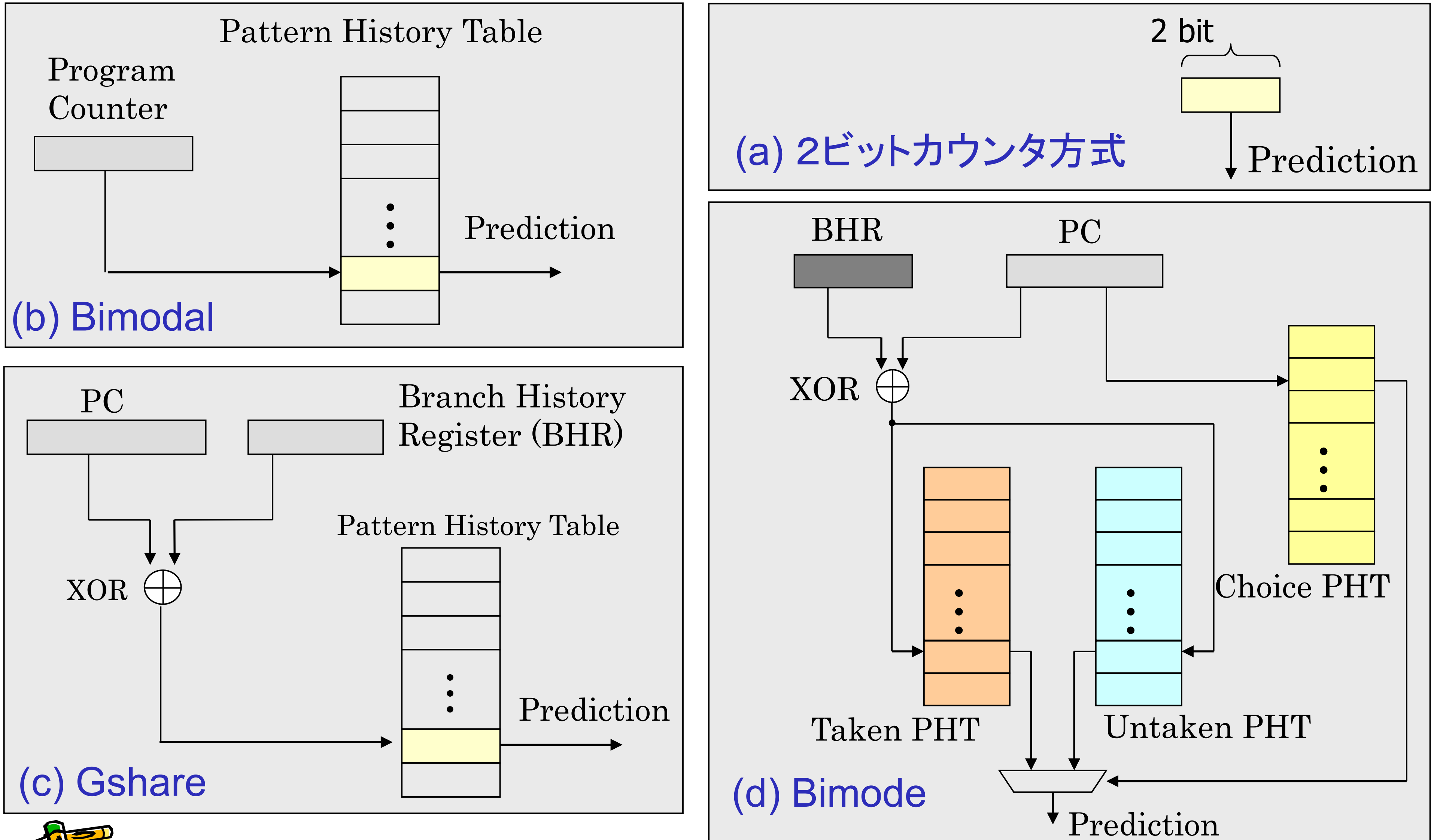


Bimode (MICRO 1997)

- 偏りを利用して競合の悪影響を緩和
  - 分岐成立に偏っているものをTaken PHTに格納
  - 分岐不成立に偏っているものをUntaken PHTに格納
- Choice PHT の内容で，どちらのテーブルを利用するか選択
- インデックスを工夫
  - Choice PHT は命令アドレス
  - Taken PHT, Untaken PHT は命令アドレスと分岐履歴

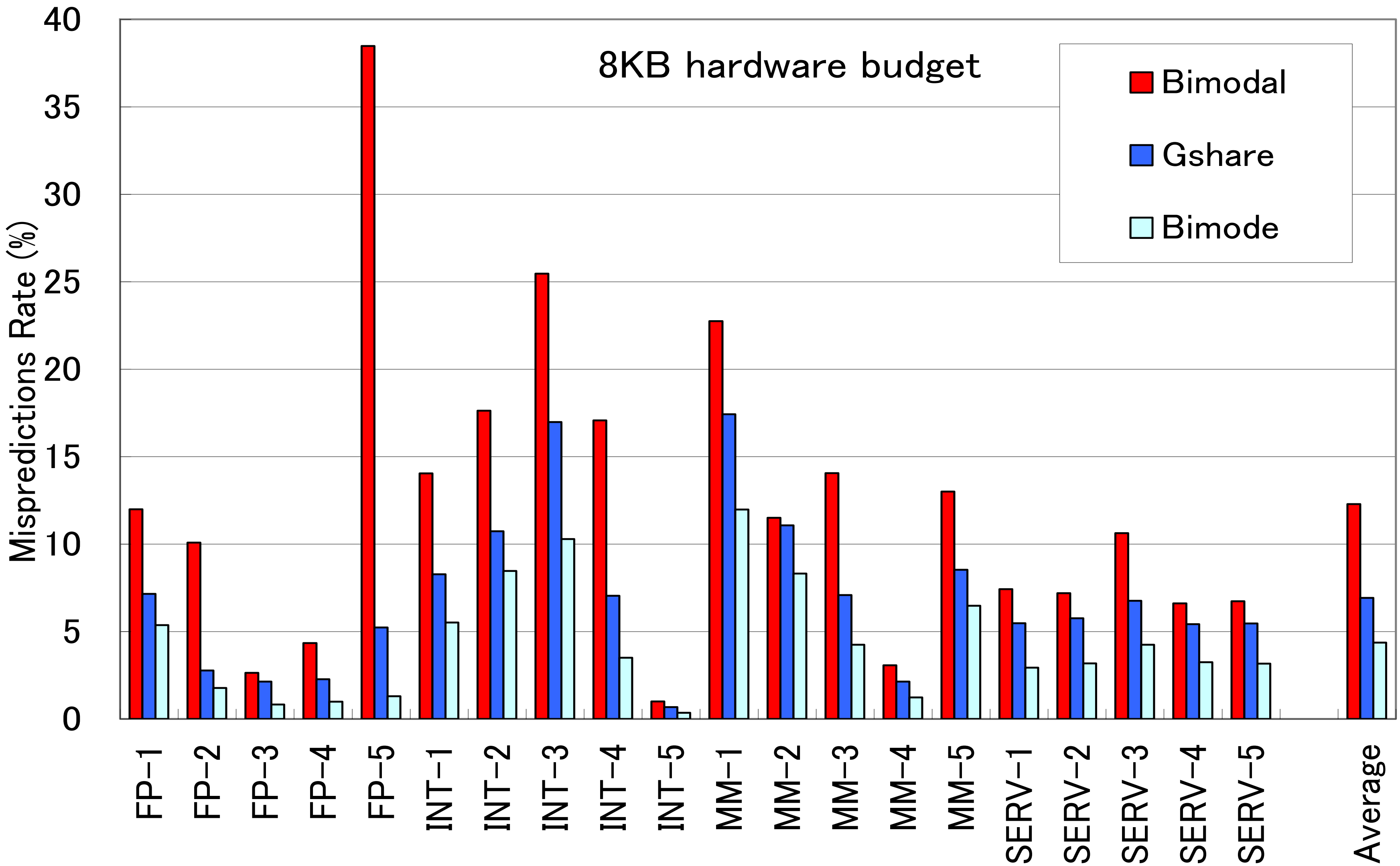


幾つかの分岐予測器



(a) 2ビットカウンタ方式
2 bit
Prediction
(b) Bimodal
Pattern History Table
Program Counter
Prediction
(c) Gshare
PC
Branch History Register (BHR)
XOR
Pattern History Table
Prediction
(d) Bimode
BHR
PC
XOR
Choice PHT
Taken PHT
Untaken PHT
Prediction

予測精度（予測ミス率）



Benchmark for CBP(2004) by Intel MRL and IEEE TC uARCH.

高いバンド幅の命令フェッチ

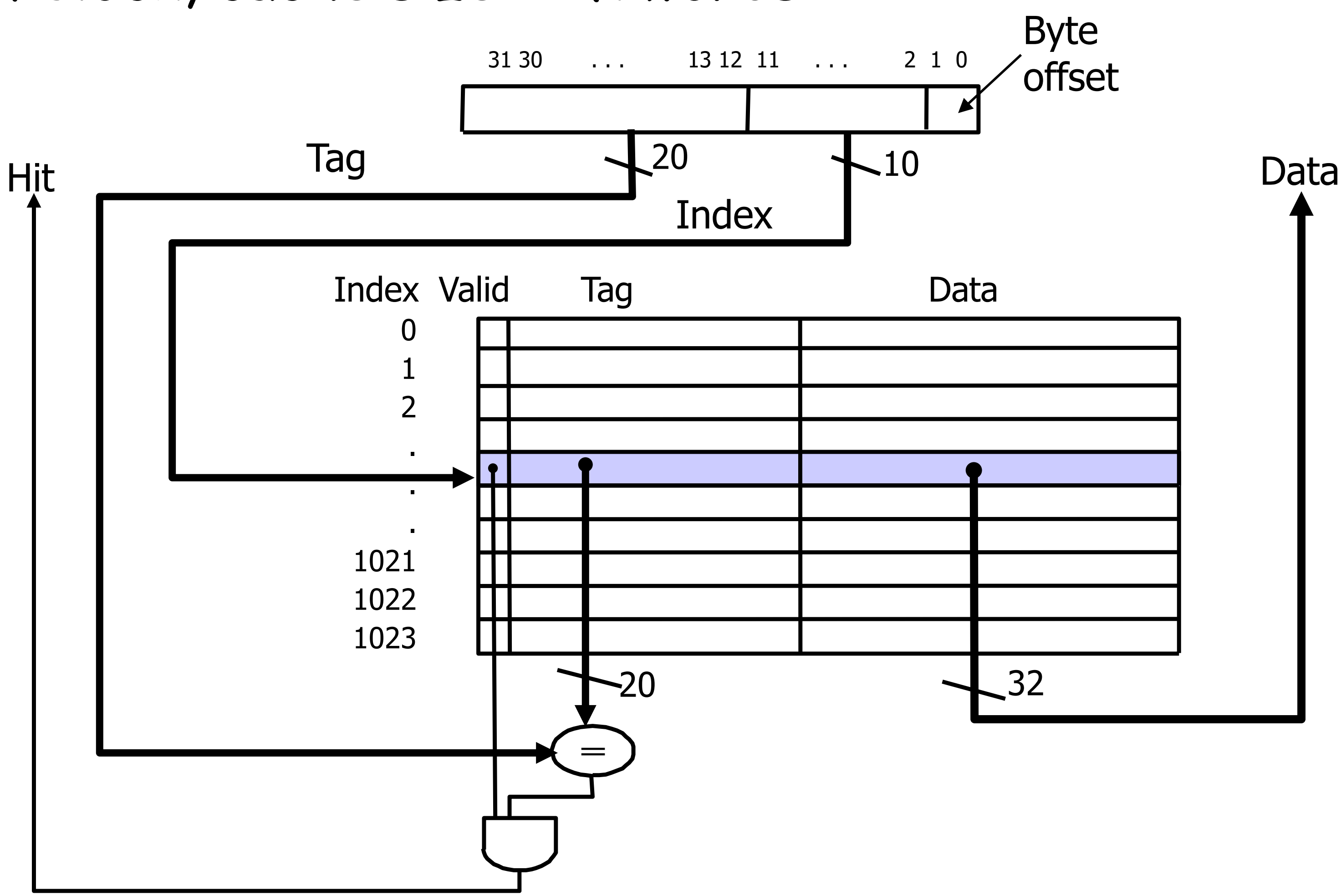
- パイプラインにバブルを生じさせないために，条件分岐命令をフェッチした時に次の3つを予測 (prediction) する.
  - フェッチしている命令が分岐命令か？
  - 分岐先アドレス(32bit)
  - 分岐方向
    - 分岐成立(Taken)か分岐不成立(Not taken, Untaken)か？
- Speculation assuming the prediction is true
  - No penalty by a branch instruction when the prediction hits
  - Recovery when the miss is detected

BTB (Branch Target Buffer)

- フェッチしている命令が分岐命令か？
- 分岐先アドレス(32bit)

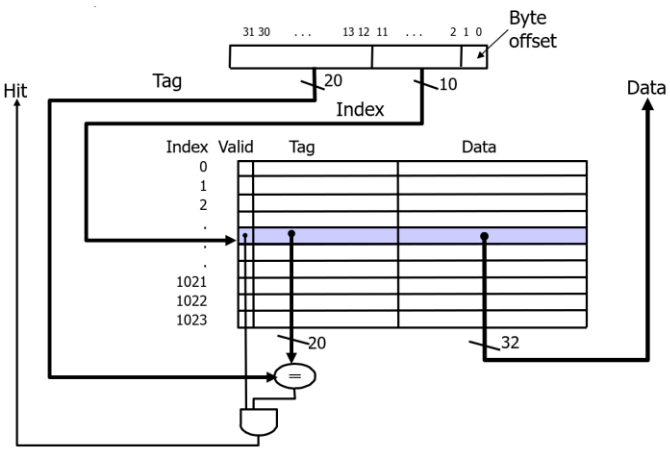
Direct Mapped Cache Example

- One word/block, cache size = 1K words

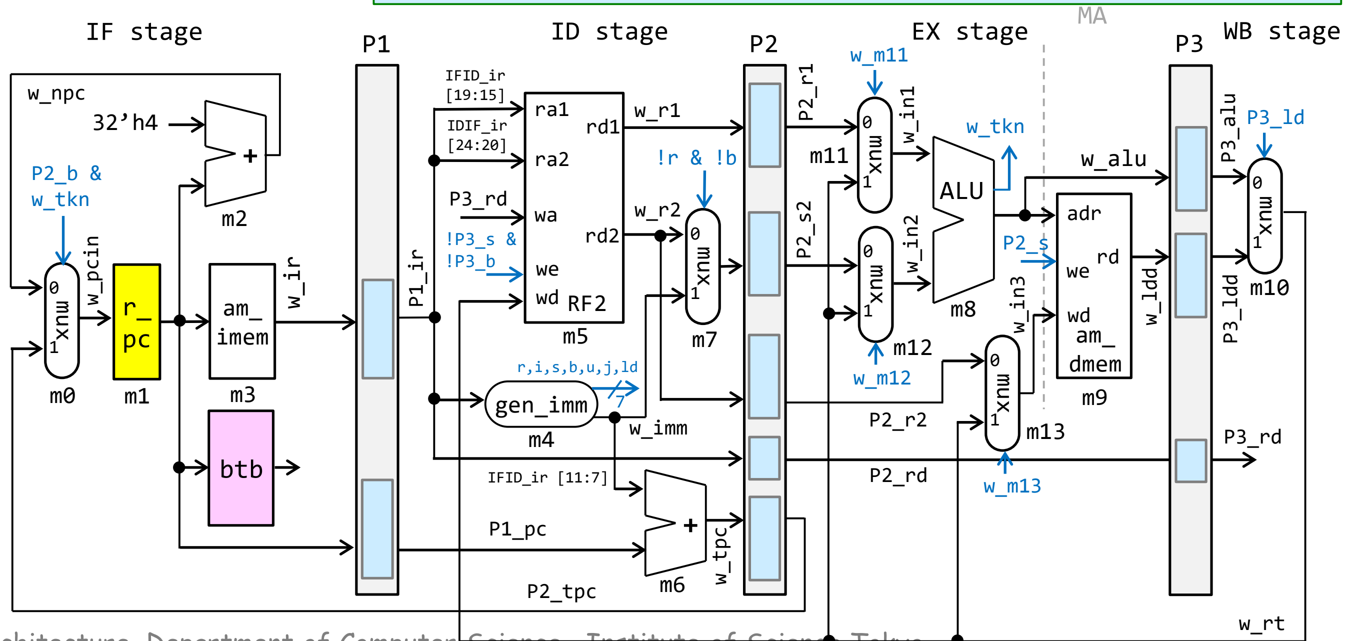


What kind of locality are we taking advantage of?

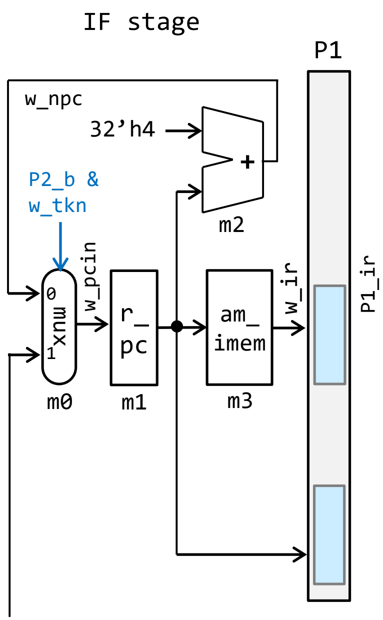
Branch Target Buffer (BTB)




```
module m_btb (
  input  wire        w_clk, w_we,
  input  wire [31:0] w_adr,
  input  wire [4:0]  w_wadr,
  input  wire [57:0] w_wd,
  output wire        w_hit,
  output wire [31:0] w_dout
);
  wire w_v;
  wire [24:0] w_tag;
  reg [57:0] mem [0:31];
  always @(posedge w_clk) if (w_we) mem[w_wadr] <= w_wd;
  assign {w_v, w_tag, w_dout} = mem[w_adr[6:2]];
  assign w_hit = w_v & (w_tag==w_adr[31:7]);
  integer i; initial for (i=0; i<32; i=i+1) mem[i] = 0;
endmodule
```



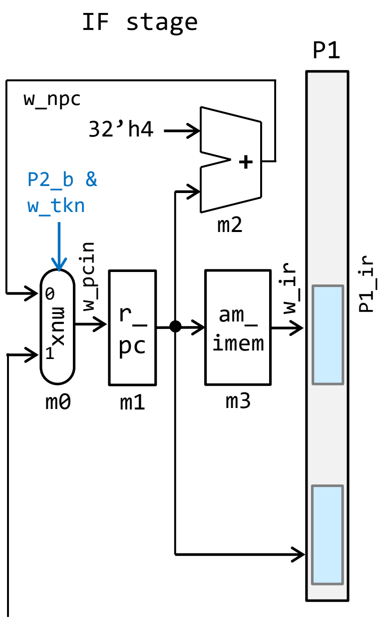
常に「分岐が成立しない」と予測.




```
module m_proc8s(w_clk);
  input wire w_clk;
  reg [31:0] P1_ir=32'h13, P1_pc=0, P2_pc=0, P3_pc=0;
  reg [31:0] P2_r1=0, P2_s2=0, P2_r2=0, P2_tpc=0;
  reg [31:0] P3_alu=0, P3_ldd=0;
  reg P2_r=0, P2_s=0, P2_b=0, P2_ld=0, P3_s=0, P3_b=0, P3_ld=0;
  reg [4:0] P2_rd=0, P2_rs1=0, P2_rs2=0, P3_rd=0;
  reg P1_v=0, P2_v=0, P3_v=0;
  wire [31:0] w_npc, w_ir, w_imm, w_r1, w_r2, w_s2, w_rt;
  wire [31:0] w_alu, w_ldd, w_tpc, w_pcin, w_in1, w_in2, w_in3;
  wire w_r, w_i, w_s, w_b, w_u, w_j, w_ld, w_tkn;
  reg [31:0] r_pc = 0; // m1
  wire w_miss = P2_b & w_tkn & P2_v;
  assign w_pcin = (w_miss) ? P2_tpc : w_npc; // m0
  assign w_npc = r_pc + 32'h4; // m2
  m_am_imem m3 (r_pc, w_ir);
  m_gen_imm m4 (P1_ir, w_imm, w_r, w_i, w_s, w_b, w_u, w_j, w_ld);
  m_RF2 m5 (w_clk, P1_ir[19:15], P1_ir[24:20], w_r1, w_r2,
            P3_rd, !P3_s & !P3_b & P3_v, w_rt);
  assign w_tpc = P1_pc + w_imm; // m6
  assign w_s2 = (!w_r & !w_b) ? w_imm : w_r2; // m7
  always @(posedge w_clk) begin
    {P1_v, P2_v, P3_v} <= {!w_miss, !w_miss & P1_v, P2_v};
    {r_pc, P1_ir, P1_pc, P2_pc} <= {w_pcin, w_ir, r_pc, P1_pc};
    {P2_r1, P2_r2, P2_s2, P2_tpc} <= {w_r1, w_r2, w_s2, w_tpc};
    {P2_r, P2_s, P2_b, P2_ld} <= {w_r, w_s, w_b, w_ld};
    {P2_rs2, P2_rs1, P2_rd} <= {P1_ir[24:15], P1_ir[11:7]};
    {P3_pc, P3_ld} <= {P2_pc, P2_ld};
    {P3_alu, P3_ldd, P3_rd} <= {w_alu, w_ldd, P2_rd};
  end
  assign w_alu = w_in1 + w_in2;    // m8
11
13
: P2_s2;// m12
```

```
`MM[0]={12'd0,5'd0,3'h0,5'd10,7'h13};        // 00   addi x10,x0,0
`MM[1]={12'd0,5'd0,3'h0,5'd3,7'h13};         // 04   addi x3,x0,0
`MM[2]={12'd101,5'd0,3'h0,5'd1,7'h13};       // 08   addi x1,x0,101
`MM[3]={7'd0,5'd3,5'd10,3'h0,5'd10,7'h33};   // 0c L:add  x10,x10,x3
`MM[4]={12'd1,5'd3,3'h0,5'd3,7'h13};         // 10   addi x3,x3,1
`MM[5]={~12'd0,5'd1,5'd3,3'h1,5'b11001,7'h63};// 14  bne  x3,x1,L
`MM[6]=32'h00050f13;                         // 18   HALT
```

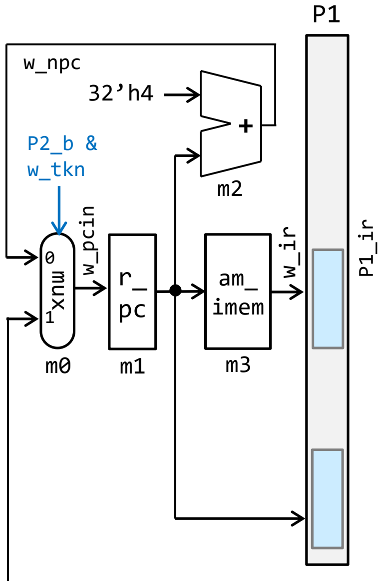
常に「分岐が成立する」と予測.




```
module m_proc8s(w_clk);
  input wire w_clk;
  reg [31:0] P1_ir=32'h13, P1_pc=0, P2_pc=0, P3_pc=0;
  reg [31:0] P2_r1=0, P2_s2=0, P2_r2=0, P2_tpc=0;
  reg [31:0] P3_alu=0, P3_ldd=0;
  reg P2_r=0, P2_s=0, P2_b=0, P2_ld=0, P3_s=0, P3_b=0, P3_ld=0;
  reg [4:0] P2_rd=0, P2_rs1=0, P2_rs2=0, P3_rd=0;
  reg P1_v=0, P2_v=0, P3_v=0;
  wire [31:0] w_npc, w_ir, w_imm, w_r1, w_r2, w_s2, w_rt;
  wire [31:0] w_alu, w_ldd, w_tpc, w_pcin, w_in1, w_in2, w_in3;
  wire w_r, w_i, w_s, w_b, w_u, w_j, w_ld, w_tkn;
  reg [31:0] r_pc = 0; // m1
  wire [31:0] w_ppc = 32'hc;
  wire w_bp_tkn = (r_pc==32'h14);
  wire [31:0] w_truepc = (P2_v & P2_b & w_tkn) ? P2_tpc : P2_pc+4;
  wire w_miss = P2_v & P2_b & P1_v & (P1_pc!=w_truepc);
  assign w_pcin = (w_miss) ? w_truepc : (w_bp_tkn) ? w_ppc : w_npc;
  assign w_npc = r_pc + 32'h4; // m2
  m_am_imem m3 (r_pc, w_ir);
  m_gen_imm m4 (P1_ir, w_imm, w_r, w_i, w_s, w_b, w_u, w_j, w_ld);
  m_RF2 m5 (w_clk, P1_ir[19:15], P1_ir[24:20], w_r1, w_r2,
            P3_rd, !P3_s & !P3_b & P3_v, w_rt);
  assign w_tpc = P1_pc + w_imm; // m6
  assign w_s2 = (!w_r & !w_b) ? w_imm : w_r2; // m7
  always @(posedge w_clk) begin
    {P1_v, P2_v, P3_v} <= {!w_miss, !w_miss & P1_v, P2_v};
    {r_pc, P1_ir, P1_pc, P2_pc} <= {w_pcin, w_ir, r_pc, P1_pc};
    {P2_r1, P2_r2, P2_s2, P2_tpc} <= {w_r1, w_r2, w_s2, w_tpc};
    {P2_r, P2_s, P2_b, P2_ld} <= {w_r, w_s, w_b, w_ld};
    {P2_rs2, P2_rs1, P2_rd} <= {P1_ir[24:15], P1_ir[11:7]};
```

```
`MM[0]={12'd0,5'd0,3'h0,5'd10,7'h13};        // 00   addi x10,x0,0
`MM[1]={12'd0,5'd0,3'h0,5'd3,7'h13};         // 04   addi x3,x0,0
`MM[2]={12'd101,5'd0,3'h0,5'd1,7'h13};       // 08   addi x1,x0,101
`MM[3]={7'd0,5'd3,5'd10,3'h0,5'd10,7'h33};   // 0c L:add  x10,x10,x3
`MM[4]={12'd1,5'd3,3'h0,5'd3,7'h13};         // 10   addi x3,x3,1
`MM[5]={~12'd0,5'd1,5'd3,3'h1,5'b11001,7'h63};// 14  bne  x3,x1,L
`MM[6]=32'h00050f13;                         // 18   HALT
```

IF stage




```
module m_proc8s(w_clk);
  input wire w_clk;
  reg [31:0] P1_ir=32'h13, P1_pc=0, P2_pc=0, P3_pc=0;
  reg [31:0] P2_r1=0, P2_s2=0, P2_r2=0, P2_tpc=0;
  reg [31:0] P3_alu=0, P3_ldd=0;
  reg P2_r=0, P2_s=0, P2_b=0, P2_ld=0, P3_s=0, P3_b=0, P3_ld=0;
  reg [4:0] P2_rd=0, P2_rs1=0, P2_rs2=0, P3_rd=0;
  reg P1_v=0, P2_v=0, P3_v=0;
  wire [31:0] w_npc, w_ir, w_imm, w_r1, w_r2, w_s2, w_rt;
  wire [31:0] w_alu, w_ldd, w_tpc, w_pcin, w_in1, w_in2, w_in3;
  wire w_r, w_i, w_s, w_b, w_u, w_j, w_ld, w_tkn;
  reg [31:0] r_pc = 0; // m1
  wire [31:0] w_ppc;
  wire [57:0] w_btb_wd = {1'b1, P2_pc[31:7], P2_tpc};
  wire w_hit;
  m_btb m14 (w_clk, r_pc, w_hit, w_ppc, P2_pc[6:2], P2_v & P2_b, w_btb_wd);
  wire w_bp_tkn = (r_pc==32'h14) & w_hit;
  wire [31:0] w_truepc = (P2_v & P2_b & w_tkn) ? P2_tpc : P2_pc+4;
  wire w_miss = P2_v & P2_b & P1_v & (P1_pc!=w_truepc);
  assign w_pcin = (w_miss) ? w_truepc : (w_bp_tkn) ? w_ppc : w_npc;
  assign w_npc = r_pc + 32'h4; // m2
  m_am_imem m3 (r_pc, w_ir);
  m_gen_imm m4 (P1_ir, w_imm, w_r, w_i, w_s, w_b, w_u, w_j, w_ld);
  m_RF2 m5 (w_clk, P1_ir[19:15], P1_ir[24:20], w_r1, w_r2,
           P3_rd, !P3_s & !P3_b & P3_v, w_rt);
  assign w_tpc = P1_pc + w_imm; // m6
  assign w_s2 = (!w_r & !w_b) ? w_imm : w_r2; // m7
  always @(posedge w_clk) begin
    {P1_v, P2_v, P3_v} <= {!w_miss, !w_miss & P1_v, P2_v};
    {r_pc, P1_ir, P1_pc, P2_pc} <= {w_pcin, w_ir, r_pc, P1_pc};
    {P2_r1, P2_r2, P2_s2, P2_tpc} <= {w_r1, w_r2, w_s2, w_tpc};
```

```
`MM[0]={12'd0,5'd0,3'h0,5'd10,7'h13};        // 00   addi x10,x0,0
`MM[1]={12'd0,5'd0,3'h0,5'd3,7'h13};         // 04   addi x3,x0,0
`MM[2]={12'd101,5'd0,3'h0,5'd1,7'h13};       // 08   addi x1,x0,101
`MM[3]={7'd0,5'd3,5'd10,3'h0,5'd10,7'h33};   // 0c L:add  x10,x10,x3
`MM[4]={12'd1,5'd3,3'h0,5'd3,7'h13};         // 10   addi x3,x3,1
`MM[5]={~12'd0,5'd1,5'd3,3'h1,5'b11001,7'h63};// 14   bne  x3,x1,L
`MM[6]=32'h00050f13;                         // 18   HALT
```

Gshare branch predictor

```
module m_gshare (
  input  wire w_clk, w_we,
  input  wire [4:0] w_wadr, w_radr,
  input  wire w_tkn,
  input  wire [4:0] w_r_brh, w_w_bhr,
  output wire w_pred
);
  reg [1:0] mem [0:31];
  wire [1:0] w_data = mem[w_radr ^ w_r_bhr];
  assign w_pred = w_data[1];

  wire [1:0] w_cnt  = mem[w_wadr ^ w_w_bhr];
  always @(posedge w_clk) if (w_we)
    mem[w_wadr ^ w_w_bhr] <= (w_cnt < 3 &  w_tkn) ? w_cnt + 1 :
                             (w_cnt > 0 & !w_tkn) ? w_cnt - 1 : w_cnt;
  integer i; initial for (i=0; i<32; i=i+1) mem[i] = 1;
endmodule
```