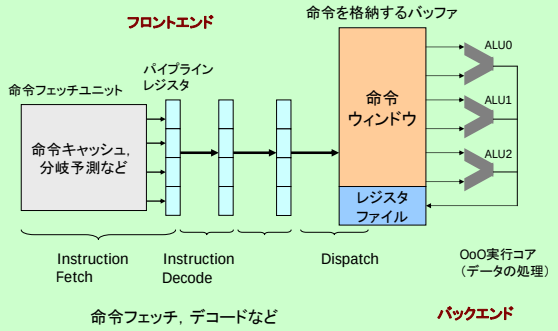
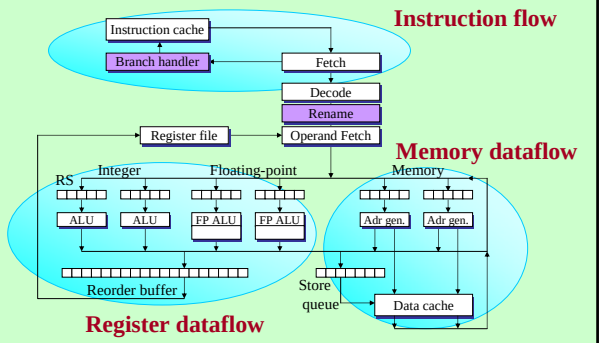
計算機アーキテクチャ特論
(Advanced Computer Architectures)

6. 分岐予測

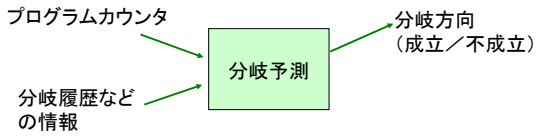
アウトオブオーダ実行プロセッサの概要



Out-of-orderスーパースカラ・プロセッサ

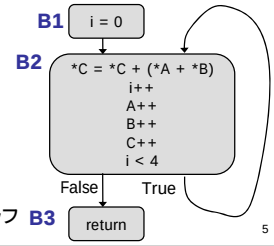


分岐方向の予測（分岐予測）

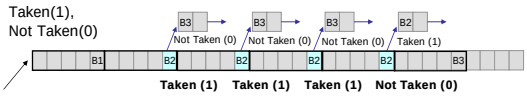


サンプルプログラム Vector Add

```
#define VSIZE 4
void vadd(long *A, long *B, long *C){
  for(i=0; i<VSIZE; i++)
    C[i] += (A[i] + B[i]);
}
```

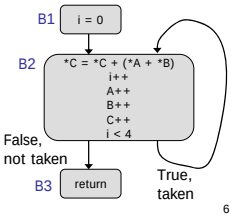


シンプルな分岐予測 Branch Always



処理すべき機械命令の列

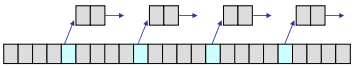
- Branch Always: 常に分岐が成立すると予測する.
  - 上の例では，予測成功率は75%, ミス率25%
  - 予測のためのメモリを必要としない.
  - 予測とよぶほどのものではない.



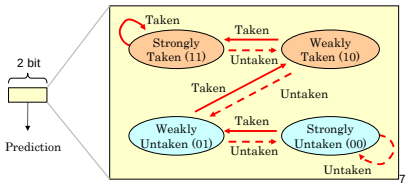
シンプルな分岐予測 2ビットカウンタ方式

トレースデータ
（分岐アドレス，分岐結果）

0040d6c5 1	0040d89c 1
0040d6b8 1	0040d89c 1
0040d6bc **0**	0040d89c 1
0040d6c5 **0**	0040d89c 1
0040d6df **0**	0040d89c 1
0040d71f **0**	0040d89c 1
0040d736 **0**	0040d89c 0
0040d7ab **0**	0040d8a2 0
0040d7cd **0**	0040d8c0 1
0040d7f9 **0**	0040d8c4 0
0040d81e 1	0040d8cd 1
0040d7f9 1	0040d8c0 0
0040d81e 1	0040d8c4 1
0040d7f9 0	0040d8cd 1
0040d81e 0	0040d8c0 1
0040d83d 0	0040d8c4 0
0040d86d **1**	0040d8cd 0
0040d86d **1**	0040d8e7 0
0040d86d **1**	0040d923 1
0040d86d **1**	0040d7ab 0
0040d86d **1**	0040d7cd 0
0040d86d **1**	0040d7f9 0



- 2ビットカウンタ方式
  - 大域的な偏り，局所性の利用
  - 2ビットカウンタの状態に応じて予測
  - 予測のためのメモリは2ビット
  - 状態の更新



Exercise

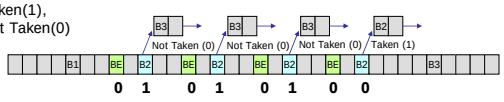
2ビットカウンタ方式の機能レベルの実装

```
class TwoBit {
    int cnt;
public:
    TwoBit(int);
    int predict();
    void update(int);
};
```

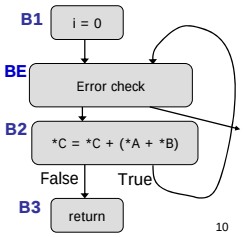
```
TwoBit::TwoBit(){
    cnt =1
}

int TwoBit::predict(){
}

void TwoBit::update(int taken){
}
```


サンプルプログラム Vector Add

Taken(1),
Not Taken(0)



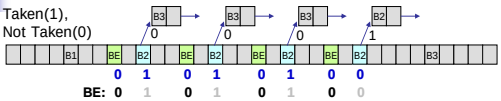

```
#define VSIZE 4
void vadd(long *A, long *B, long *C){
  for(i=0; i<VSIZE; i++) {
    if(A[i]<0) error_routine();
    C[i] += (A[i] + B[i]);
  }
}
```



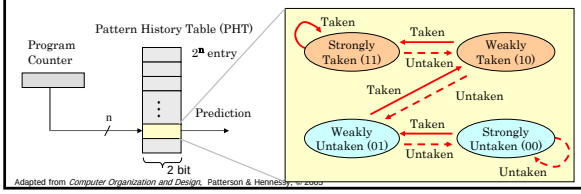
制御フローグラフ

Bimodal (ISCA 1981)

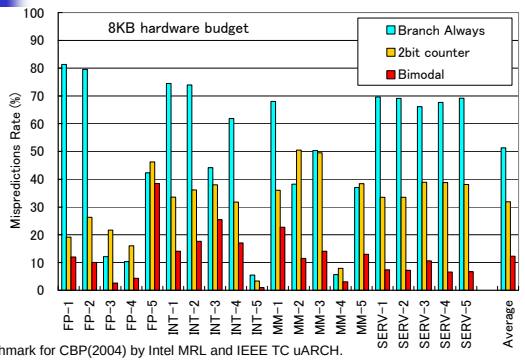
Taken(1),
Not Taken(0)



- 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．

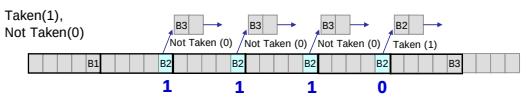


シンプルな分岐予測の予測精度（予測ミス率）



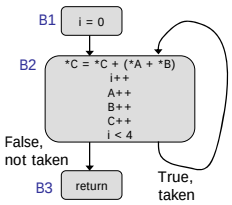
Benchmark for CBP(2004) by Intel MRL and IEEE TC uARCH.

分岐履歴

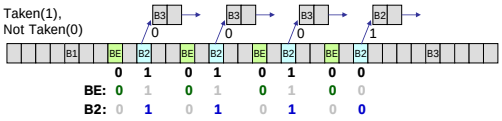


B2の分岐履歴

1110 ?
11101 ?
111011 ?
1110111 ?
11101110 ?



ローカル，グローバル分岐履歴



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

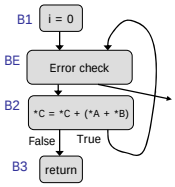
1110 ?
11101 ?
111011 ?
1110111 ?
11101110 ?

ローカル分岐履歴

B2とBEの分岐履歴

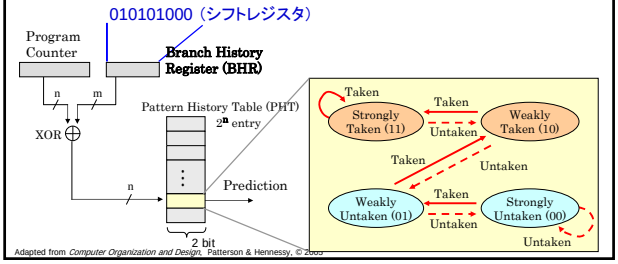
010101000 ?

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の機能レベルの実装

```
typedef int data_t;

class Gshare {
    data_t bhr;
    data_t *buf;
public:
    int size;
    Gshare(int);
    int predict(data_t);
    void update(data_t, int);
};
```

```
Gshare::Gshare(int bpred_size){
    size = bpred_size;
    buf  = (data_t *)calloc(size, sizeof(data_t));
    for(int i=0; i<size; i++) buf[i] = 2;
}

int Gshare::predict(data_t pc){

}

void Gshare::update(data_t pc, int taken){

}
```

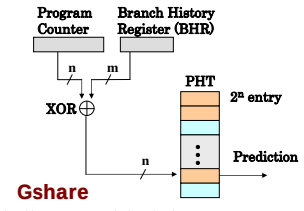

Gshareの機能レベルの実装

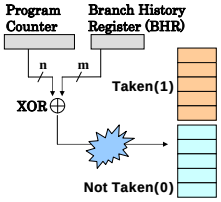
```
typedef int data_t;

class Gshare {
    data_t bhr;
    data_t *buf;
public:
    int size;
    Gshare(int);
    int predict(data_t);
    void update(data_t, int);
};
```

```
Gshare::Gshare(int bpred_size){
    size = bpred_size;
    buf  = (data_t *)calloc(size, sizeof(data_t));
    for(int i=0; i<size; i++) buf[i] = 2;
}

int Gshare::predict(data_t pc){
    int index = ((pc >> 2) ^ bhr) % size;
    return (buf[index]>1);
}

void Gshare::update(data_t pc, int taken){
    int index = ((pc >> 2) ^ bhr) % size;
    if(taken!=0 && buf[index]<2) buf[index]++;
    if(taken==0 && buf[index]>0) buf[index]--;
    bhr = (bhr << 1) | taken;
}
```


Gshare の改良

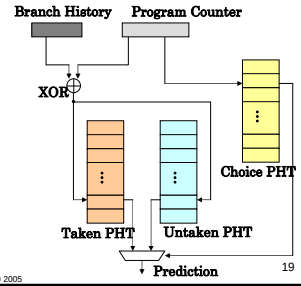
- PHTの競合が発生して性能が低下
- PCとBHRによって特定される予測（成立，不成立）には偏りが存在するので，これらを別のテーブルに格納することで競合の悪影響を緩和
  - 分岐成立に偏っているもの
  - 分岐不成立に偏っているもの



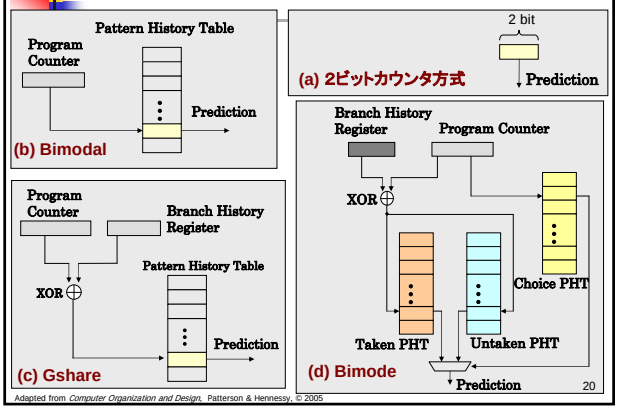


Bimode (MICRO 1997)

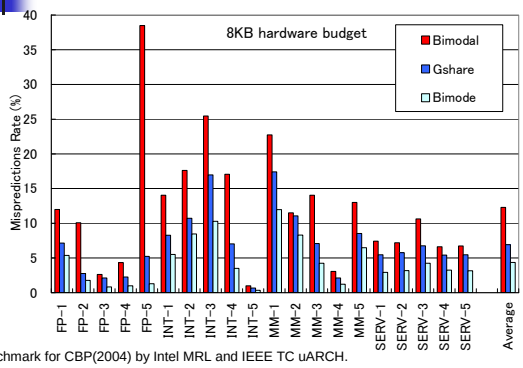
- 偏りを利用して競合の悪影響を緩和
  - 分岐成立に偏っているものをTaken PHTに格納
  - 分岐不成立に偏っているものをUntaken PHTに格納
- Choice PHT の内容で，どちらのテーブルを利用するか選択
- インデックスを工夫
  - Choice PHT は命令アドレス
  - Taken PHT, Untaken PHT は命令アドレスと分岐履歴



ここまでの分岐予測器



予測精度（予測ミス率）



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