Branch Predictors

- Branch predictors are specialized digital circuits inside modern CPUs that guess the outcome of conditional branches (like if-then-else statements) before the actual result is known.
- Their primary goal is to keep the CPU's instruction pipeline full and avoid costly stalls by speculatively executing the most likely path.
- Without prediction, the CPU would need to wait for each branch decision to be fully resolved, potentially wasting 10-20 clock cycles in modern processors with long pipelines.

Branch Prediction Strategies

Static Prediction

Uses simple fixed rules without history:

- Always predict branch not taken
- Predict backward branches (loops) as taken
- Predict based on branch opcode

Advantages: No hardware tables needed, power efficient

Dynamic Prediction

Uses runtime history stored in hardware tables:

- Two-bit saturating counters track branch behavior
- Pattern history tables remember sequences
- Tournament predictors combine multiple strategies

Advantages: Higher accuracy, adapts to program behavior

Advanced Techniques

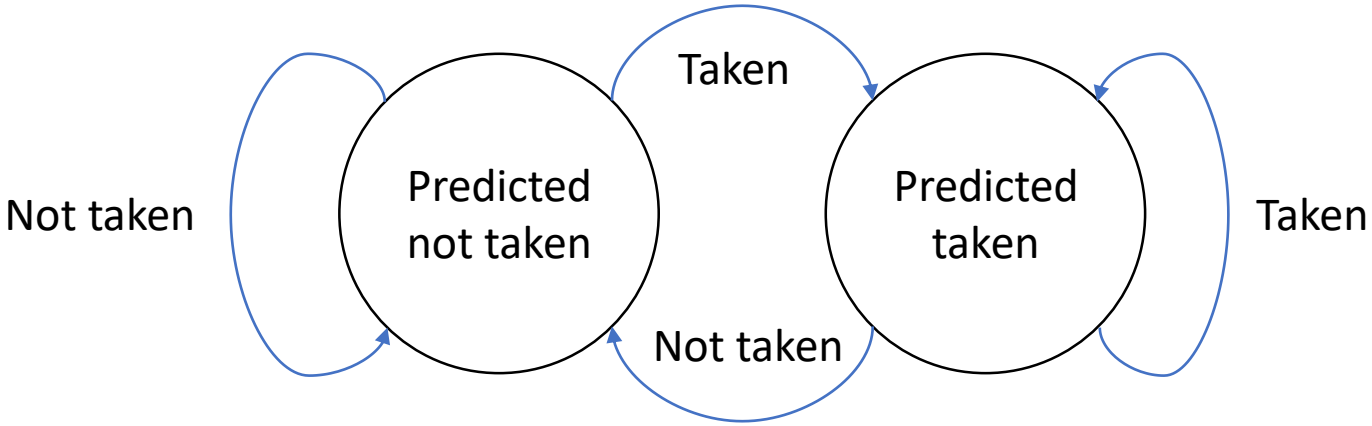
Modern CPUs combine multiple approaches:

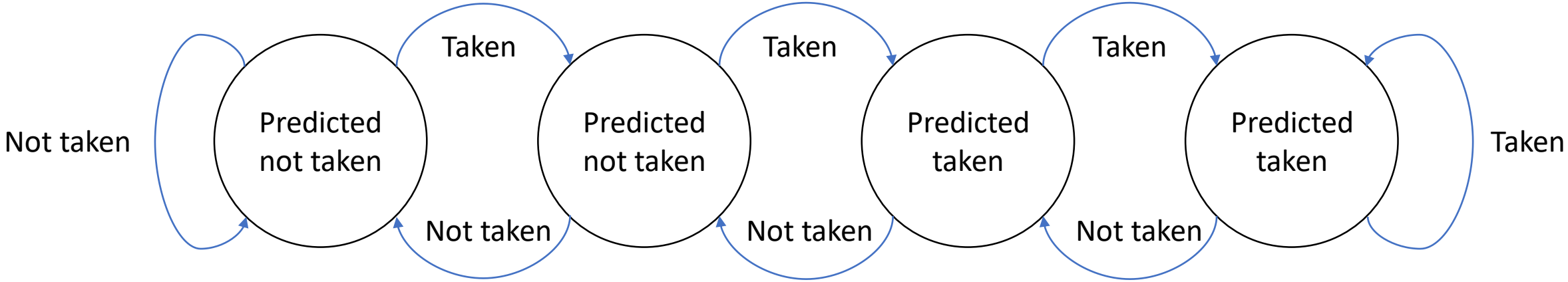
- Branch target predictors (where to jump)
- Neural network predictors (AMD, Apple)
- TAGE predictors with multiple history lengths

The better the predictor, the fewer pipeline stalls and higher CPU efficiency

Branch prediction continues to evolve with each CPU generation, with designers constantly seeking the optimal balance between accuracy, power consumption, and silicon area. Modern predictors can achieve over 95% accuracy on most workloads.

Dynamic Predictors





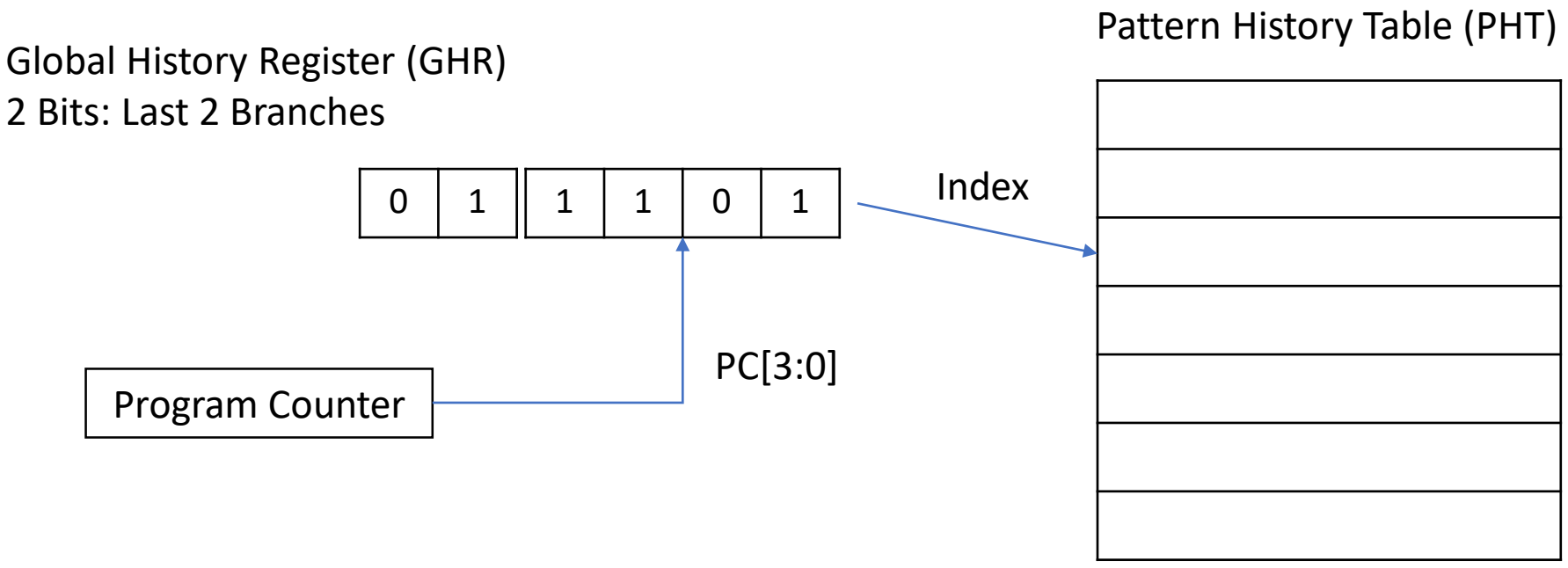
Branch Correlation

- Basic 2-bit predictors use only the recent behavior of a single branch, missing important correlations between different branches in the code.

```
if (aa==2)
    aa=0;
if (bb==2)
    bb=0;
if (aa!=bb) {
}
```

```
        DADDIU R3,R1,#-2
        BNEZ   R3,L1          ;branch b1 (aa!=2)
        DADD   R1,R0,R0       ;aa=0
L1:     DADDIU R3,R2,#-2
        BNEZ   R3,L2          ;branch b2 (bb!=2)
        DADD   R2,R0,R0       ;bb=0
L2:     DSUBU  R3,R1,R2       ;R3=aa-bb
        BEQZ   R3,L3          ;branch b3 (aa==bb)
```

Two-Level Predictors



Example

```
if (aa==2)
    aa=0;
if (bb==2)
    bb=0;
if (aa!=bb) {
```

```
        DADDIU R3,R1,#-2
0x04:   BNEZ   R3,L1          ;branch b1 (aa!=2)
        DADD   R1,R0,R0       ;aa=0
L1:     DADDIU R3,R2,#-2
0x10:   BNEZ   R3,L2          ;branch b2 (bb!=2)
        DADD   R2,R0,R0       ;bb=0
L2:     DSUBU  R3,R1,R2       ;R3=aa-bb
0x1C:   BEQZ   R3,L3          ;branch b3 (aa==bb)
```

GHR

0	0	1	1	0	0
0	1	1	1	0	0
1	0	1	1	0	0
1	1	1	1	0	0

→ Always Taken

Intel Core i7 Branch Predictor

Two-Level Design

Smaller first-level predictor for single-cycle prediction, larger second-level predictor as backup

Combined Approach

Each level combines three predictors:

- Simple two-bit predictor
- Global history predictor
- Loop exit predictor (counts iterations)

Additional Units

Separate predictors for indirect branches and return addresses