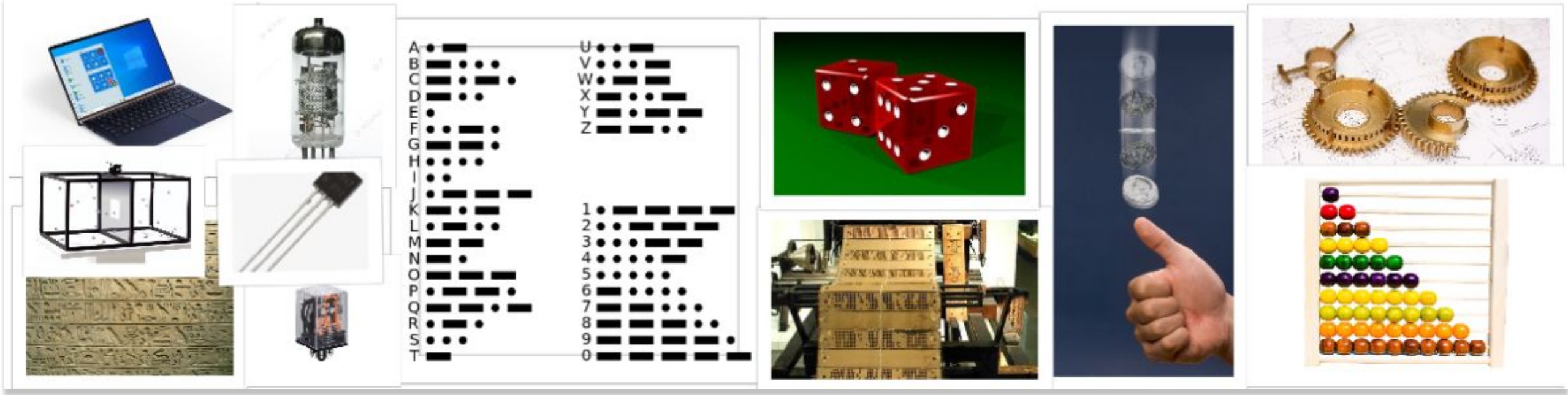


Computer Architecture

Design and Analysis of Instructions

Minimization of Control Hazards in Pipelined MIPS (RISC) Processor

Branch Correlation

- In the execution path, relation among the recently executed branch outcome and other branch instruction

- if (condition1) {…}
- …
- if (condition1 && condition2){…}

- What about not taken in the first case?

Branch Correlation

- In the execution path, relation among the recently executed branch outcome and other branch instruction

- if (condition1){a=2;}
- …
- if (a==0){…}

- What about taken in the first case?

Branch Correlation

- In the execution path, relation among the recently executed branch outcome and other branch instruction

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

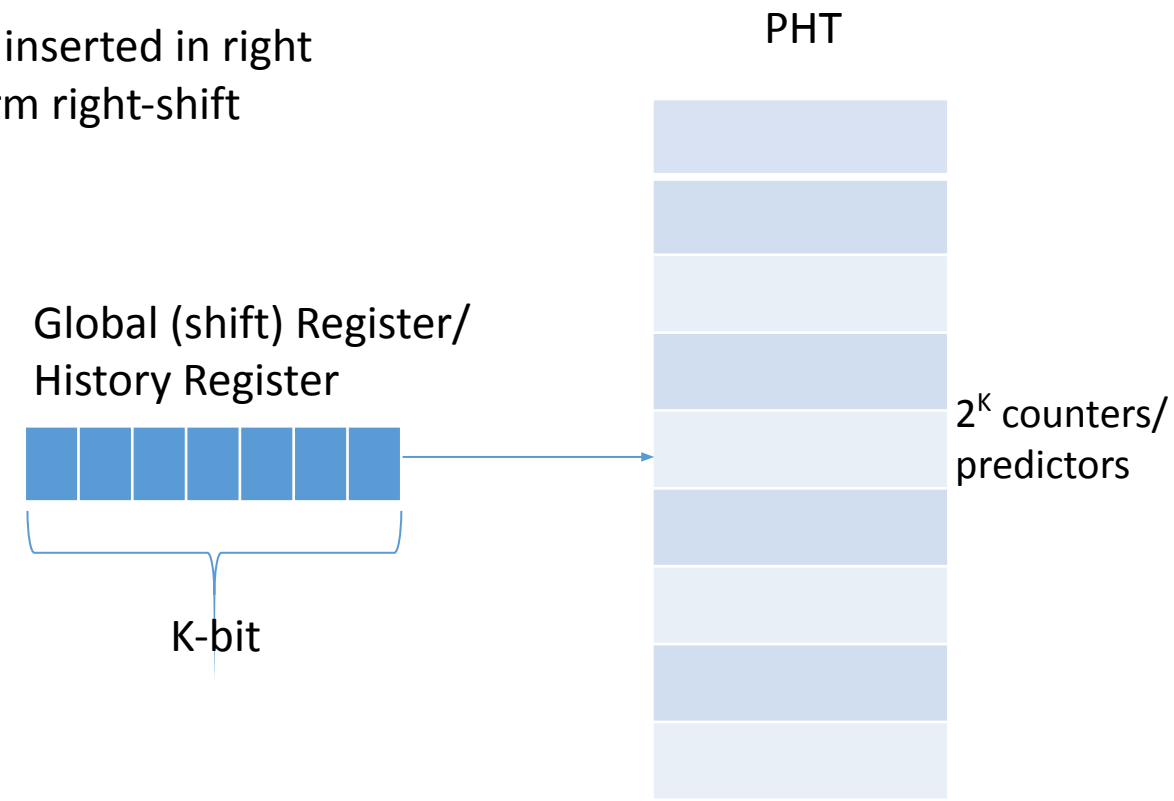
- What about taken in the first and second case?

Correlated branch predictor

```
if (aa==2){aa=0;} //b1
if (bb==2){bb=0;}//b2
if (aa!==bb){…}  //b3
```

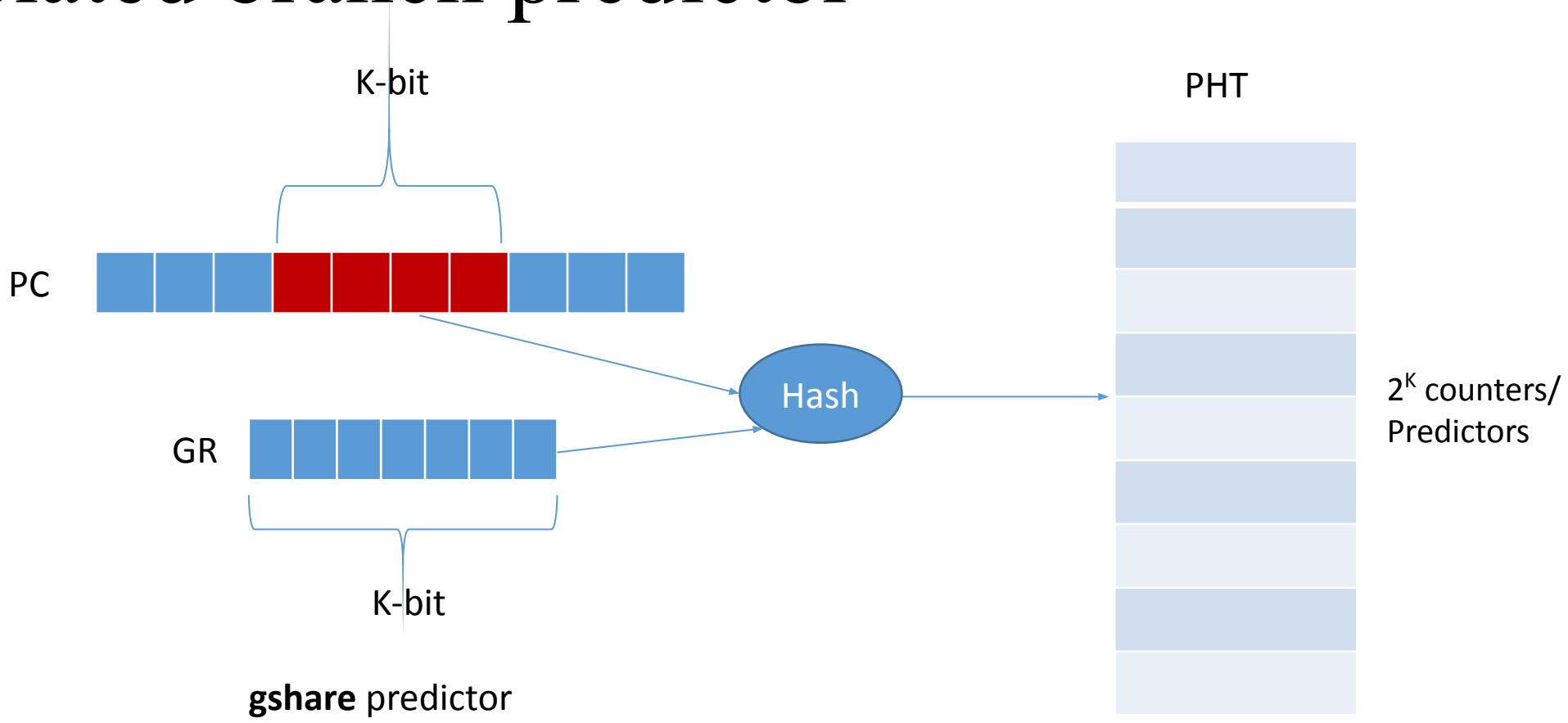
b3	b2	b1
0	0	0
0	0	1
0	1	0
	.	
	.	

Branch outcome (1/0) is inserted in right
Insert in MSB and perform right-shift



- No consideration of position of the branch in the program

Correlated branch predictor



- Hash (XOR) bits of PC and of GR to form the index

Two-level branch predictor

- *gshare* predictor introduces some locality in the indexing process, it is not sufficient for deducing pattern of individual branches
- n=4, bimodal predictor mispredict at the end of the inner loop one out of five times and global predictor might mispredict it from zero to five times depend on the branches and their outcome inside the loop
- What if local history register associated with the branch, at the end of the loop, the pattern of "5 taken, 1 not taken" would be recognized after a training period and prediction would be correct after that
- First level of two-level predictor more attuned to the locality of the branches and replace the single register with the table of local history register (LHT)

```
for (i=0; i<m; i++)
    for(j=0; j<= n; j++)
     {….}
```

Problem & solution of 2-bit branch predictor

```
for (i=0; i<m; i++)
      for(j=0; j <= 4; j++)
       {….}
```

Problem with 2-bit predictor

Local history	Outcome	States	Predictors	Initial States	
J=0	T(1)	000000	P^0	00	1x
J=1	T(1)	100000	P^1	00	1x
J=2	T(1)	110000	P^2	00	1x
J=3	T(1)	111000	P^3	00	1x
J=4	T(1)	111100	P^4	00	1x
J=5	NT(0)	111110	P^5	00	0x

What if states' encoding bits are less than number of history

Two-level branch predictor

Another motivational example

```
a=0; b=5;
While (a<1000){
 B1:     if(a%2==0){…}
         a++;
         .
         .
 B2:    if(b==0){…}
}
```

Branch History
B1: TNTNTNTN
B2: NNNNNNN

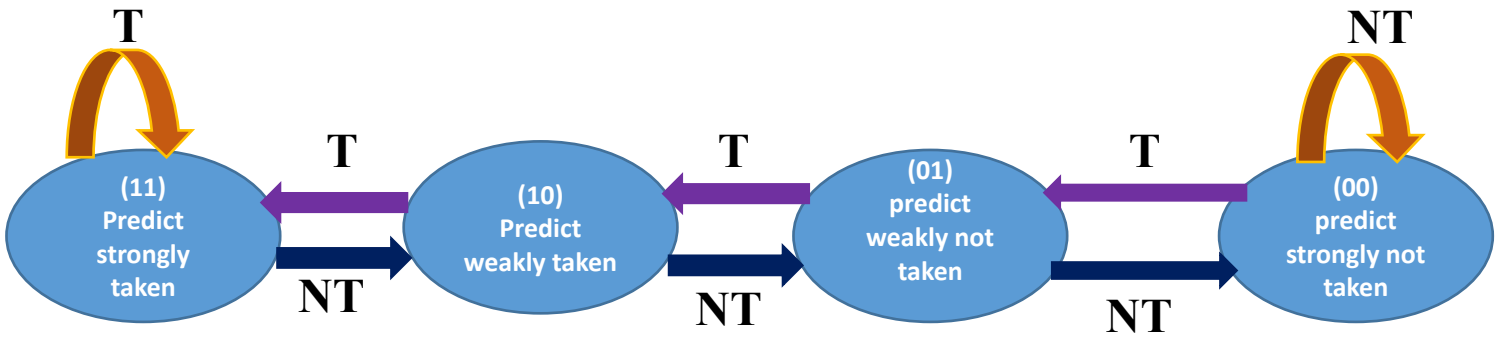
- What if a program contains 10,000 braches?
- Share the FSM
- How does one share the FSM?

- B1 is not predicting the outcome properly.
- Learn the information from the other branch, so that B1's prediction can be improved
- How does one do that?

(3,2) Correlating/Two-level Branch Prediction technique

```
a=0; b=5;
While (a<1000){
 B1:
if(a%2==0){…}
          a++;
          .
          .
 B2:    if(b==0){…}
}
```

Branch History
B1: TNTNTNTN
B2: NNNNNNN



The saturation up-down counter scheme

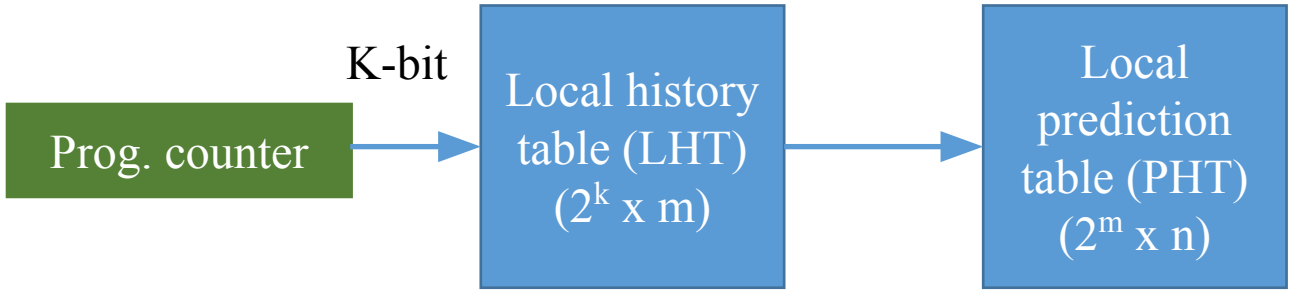
We can use other scheme, also.

See the example in the 2nd tab of AdvBranchPredictor.xlxs

What is the shortcoming of this approach?

Two-level Branch Prediction

- (m, n) predictor uses the behavior of the last m-branches to choose from 2^m branch predictors, each of which is an n-bit predictor for a single branch



- Updating: insert the branch outcome into the MSB of LHT and perform right-shift, pointed by the index register (PC)

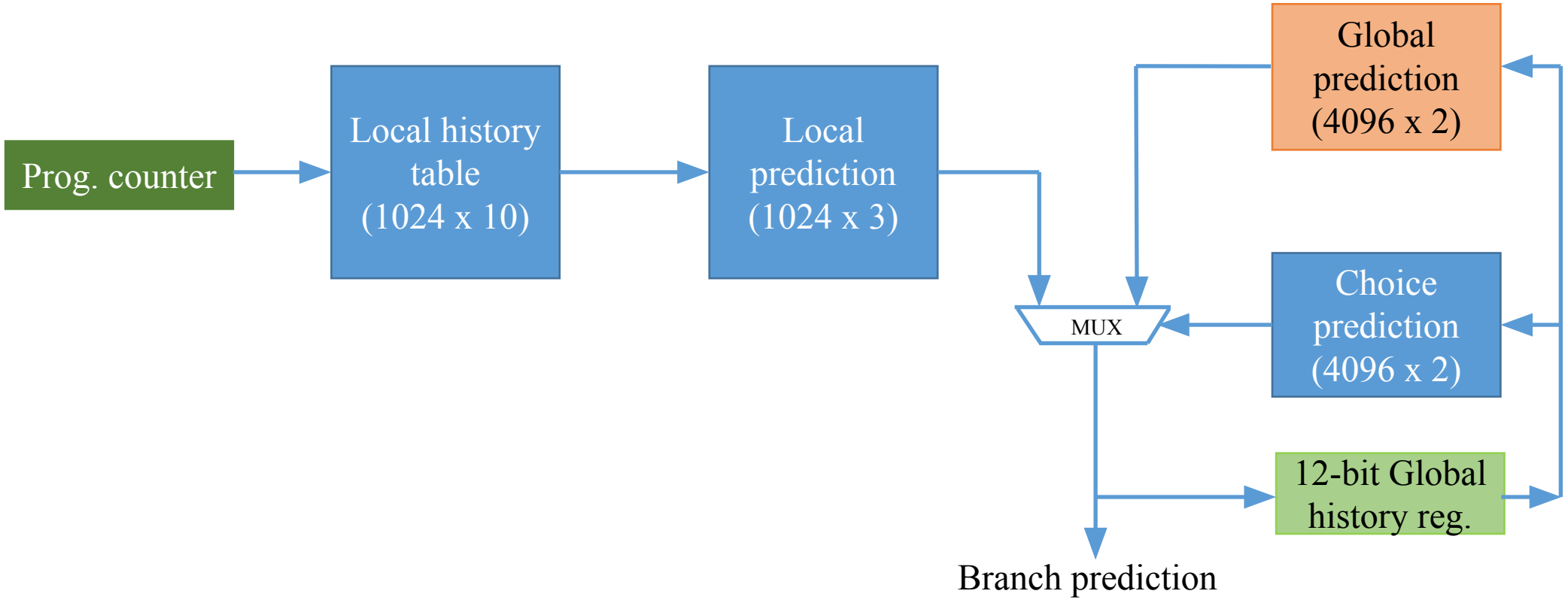
Why Tournament Branch Predictor?

- In correlated branch predictor, each entry in the local history table keep track of the outcomes of all the branches, which are mapped to that particular entry
- It cannot exploit any information about the branches mapped onto a different entry in the local history table
  - Different entries in the LHT use the same branch predictor from PHT
    - What if they have opposite branch decision
- How does one exploit such information?
- Using the Tournament branch predictor

Tournament Branch Predictor

- It consider the local history table as well as the global history table.
- Local history table keep track of the outcomes of all the branches which are mapped to a single entry
- Global history table keep track of the outcome of all the branches that are executed so far
- This is implemented in “Alpha 21264” processor and after that there are several other processors are also used the variant of this tournament branch predictor mechanism.

Dynamic branch prediction:
Organization of a Tournament Branch Predictor



Organization of a Tournament Branch Predictor

- Global history register keeps of history of the branch decision of last few branch occur in the program execution
- Here it is 12 number of branches. It can be same branches or different branches.
- Whereas the local history table keeps the history of the last few branch decision of the same branch.

Dynamic branch prediction:

Tournament Branch Prediction Rules

Local prediction table:

- Predict “taken” if the val(3-bit counter) ≥ 4

Global prediction table:

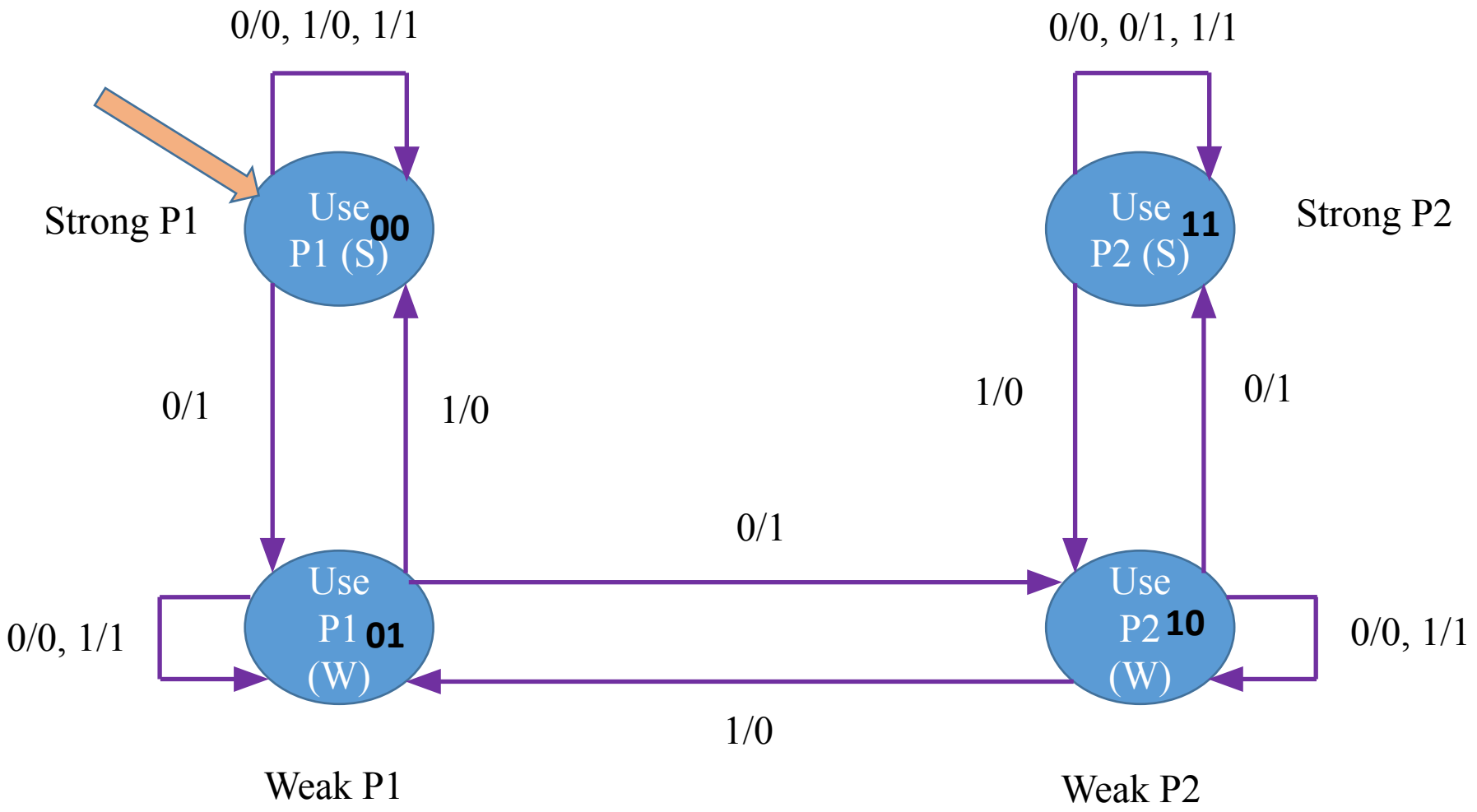
- Predict “taken” if the val(2-bit counter) ≥ 2

Prediction is made if both the local and global predictions are same.
Otherwise, consult with the choice prediction

Choice prediction:

- Do not modify if both prediction are same
- Modify the state towards the correct predictor

State diagram of Tournament (choice) predictor



1: Prediction is correct [*]
0: Prediction is incorrect
m/n: Prediction for P1/P2

[*] We will say the prediction is correct if it matches the actual outcome

Dynamic branch prediction:

Tournament Branch Prediction Rules

Update the local history table:

- Push the decision into the MSB of LHT

Update the Global history register:

- Push the decision into the MSB of GHR

Update the Local and Global prediction table:

- Based on the n-bit saturation counter

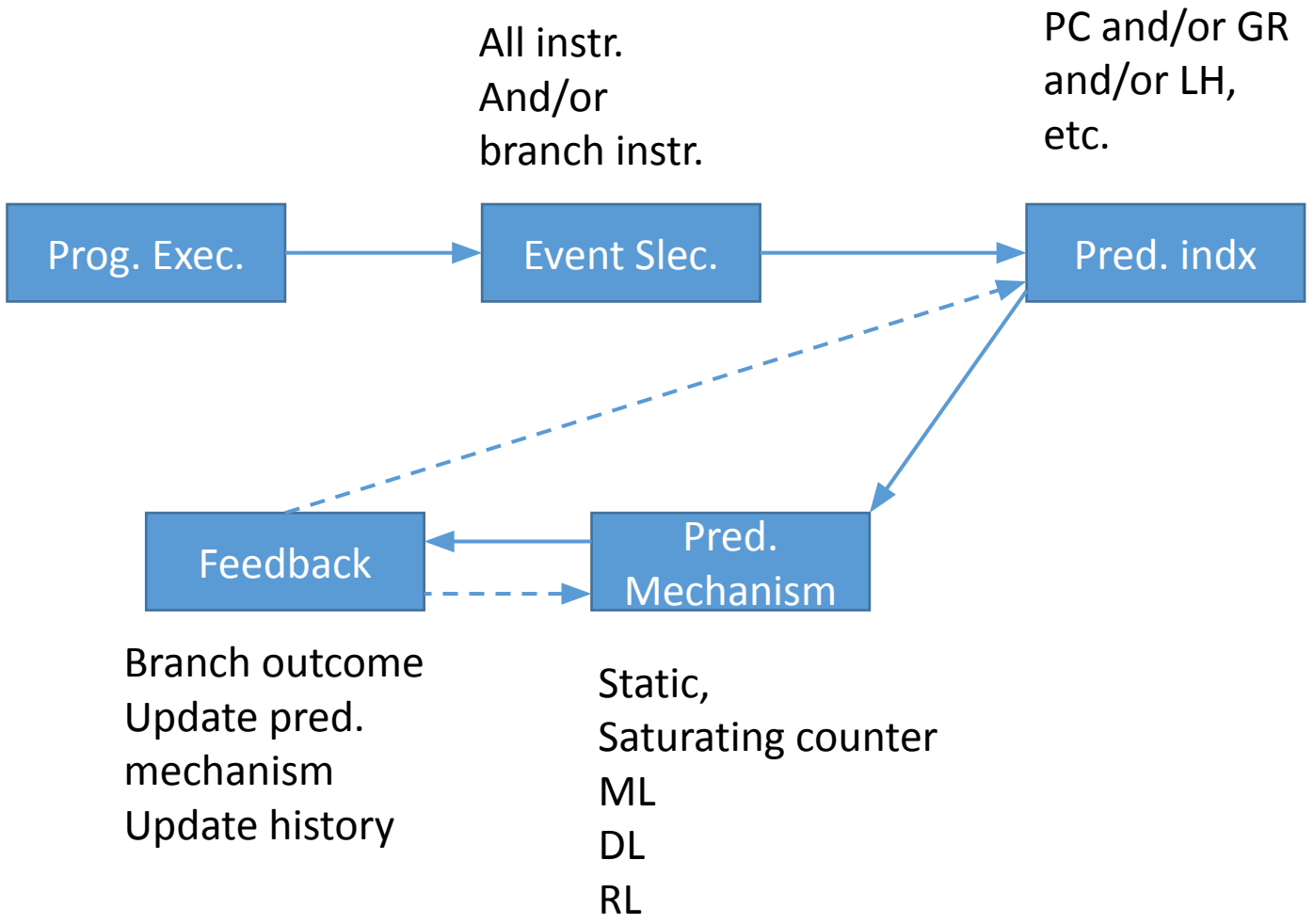
Update the Choice table

- Follow the state diagram

Dynamic branch prediction:

Tournament Branch Prediction Technique is explained using an example, in xlxs file. Please refer to the AdvBranchPredictor.xlxs file.

Overall view of branch predictor



How to initialize the predictor:

- All 0's
- All 1's
- Randomly with a fair amount of time to fill the predictor
- Some instruction sets contain a bit whether as associated branch is expected to taken or not
- Set the initial prediction based on branch direction: forward going direction (not taken) and backword direction (taken)[loop]

Sophisticated BP & other info:

- Researcher are using NN/ML/DL/RL techniques
- Championship Branch Prediction (CBP)
  - Intel
  - https://hpca23.cse.tamu.edu/taco/camino/cbp2/index.html
  - ChampSim Simulator

Improve the Branch misses

- A good example:
  - Perf tool: https://wiki.eclipse.org/Linux_Tools_Project/PERF/User_Guide
  - VTune

```
cclab@Z1-055:~/Desktop/2020B3A71516G_CA_Lab8/Perf$ perf stat -e branch-misses ./a.out

 Performance counter stats for './a.out':

      15,51,79,700      branch-misses

       5.116569727 seconds time elapsed

       5.116619000 seconds user
       0.000000000 seconds sys
```

```
4   long rand_partsum(int n)
5   {
6     int i,k;
7     long sum = 0;
8     int *vec = (int*) malloc(n * sizeof(int));
9
10    for (i = 0; i < n; i++)
11      vec[i] = rand()%n;
12
13    for (k = 0; k < 1000000; k++)
14      for (i = 0; i < n; i++)
15        if (vec[i] > n/2)
16          sum += vec[i];
17    return sum;
18  }
```

Improve the Branch misses

- A good example:
  - Perf tool: https://wiki.eclipse.org/Linux_Tools_Project/PERF/User_Guide
  - Vtune
  - Macro

```c
#include <stdio.h>
#include <stdlib.h>
#define unlikely(expr) __builtin_expect(!!(expr), 0)
#define likely(expr) __builtin_expect(!!(expr), 1)
```

```c
long rand_partsum(int n)
{
    int i,k;
    long sum = 0;
    int *vec = (int*) malloc(n * sizeof(int));

    for (i = 0; i < n; i++)
        vec[i] = rand()%n;

    for (k = 0; k < 1000000; k++)
        for (i = 0; i < n; i++)
            if (vec[i] > n/2)
                sum += vec[i];
    return sum;
}
```

Improve the Branch misses

- A good example:
  - Perf tool: https://wiki.eclipse.org/Linux_Tools_Project/PERF/User_Guide
  - Vtune
  - Branchless programming

```
13  for (k = 0; k < 1000000; k++)
14    for (i = 0; i < n; i++){
15      sum += (vec[i] > n/2) * vec[i];
16    }
17  return sum;
```

```
4   long rand_partsum(int n)
5   {
6     int i,k;
7     long sum = 0;
8     int *vec = (int*) malloc(n * sizeof(int));
9
10    for (i = 0; i < n; i++)
11      vec[i] = rand()%n;
12
13    for (k = 0; k < 1000000; k++)
14      for (i = 0; i < n; i++)
15        if (vec[i] > n/2)
16          sum += vec[i];
17    return sum;
18  }
```

https://johnnysswlab.com/how-branches-influence-the-performance-of-your-code-and-what-can-you-do-about-it/

https://blog.cloudflare.com/branch-predictor/

https://www.infoq.com/articles/making-code-faster-taming-branches/

Summary:

- Shortcomings of 2-bit predictor
- Correlated branch prediction
  - gshare branch predictor
  - Two-level branch predictor
  - Shortcoming of Correlated predictor
- Tournament predictor
- How to initialize the branch predictor's initial state
- Sophisticated branch predictor
- Championship Branch Prediction (CBP)
- Reduction techniques for branch misses