Assignment 1: Branch Predictor Implementation

In this assignment, you will explore the effectiveness of branch direction prediction (taken vs not taken) and branch target prediction using an actual program. Your task is to use the given branch prediction simulation infrastructure to evaluate the effectiveness of some simple branch prediction schemes. The simulation infrastructure can be downloaded from the class website (cbp2-infrastructure-v2.tar).

To do this, you'll implement a C++ branch prediction class (see readme contained in the tar file) that reads in the trace and simulates a Pentium M dynamic branch predictor. The Pentium M is a family of mobile 32-bit single-core x86 microprocessors specifically designed for mobility (notebook computers). Improved branch prediction was designed for increased performance as well as for reduced power consumption. The article "*Experiment Flows and Microbenchmarks for Reverse Engineering of Branch Predictor Structures"* discloses architecture details of Pentium M branch predictor with diagram below. In this homework, you will implement part of the Pentium M branch predictors used for branch direction prediction.

Figure. Pentium M CPU die. **Can you spot the branch predictors?**

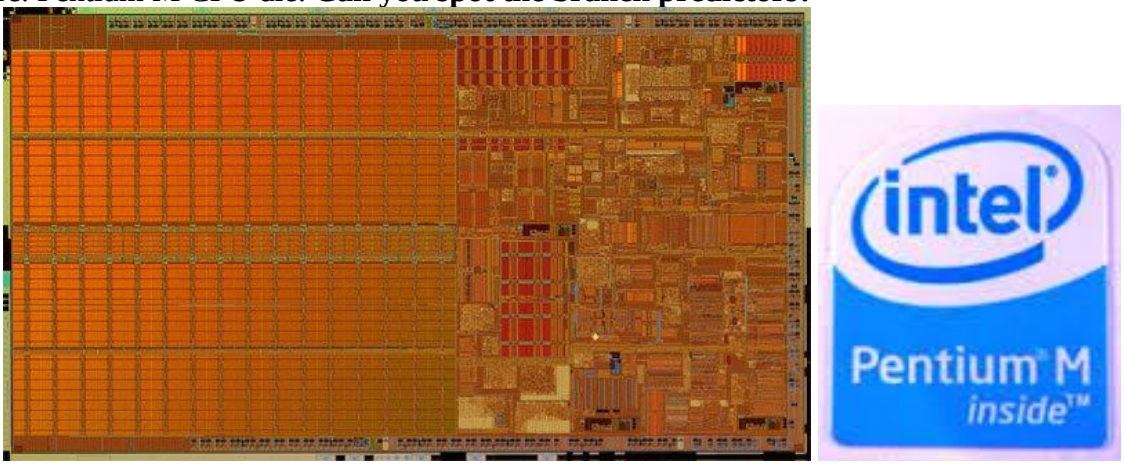
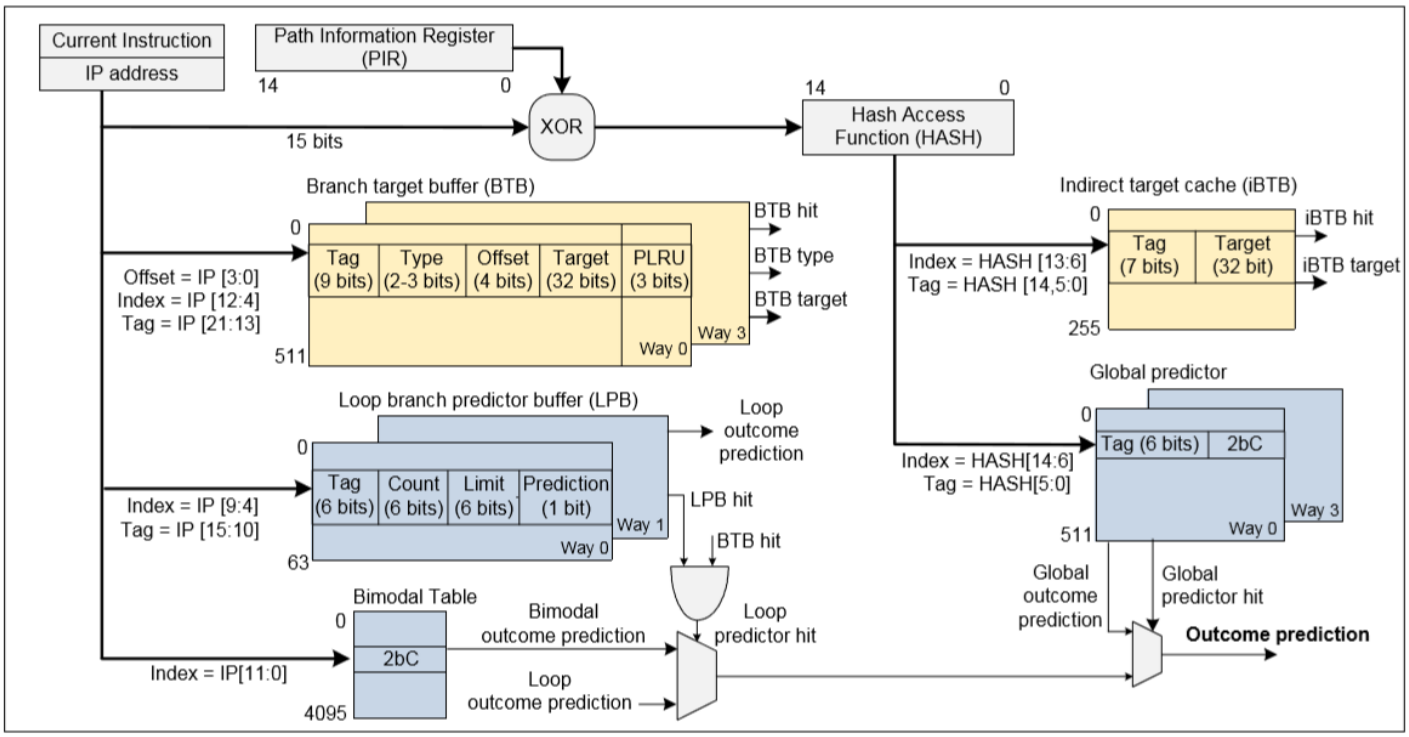


Figure. Pentium M branch predictors (*Experiment Flows and Microbenchmarks for Reverse Engineering of Branch Predictor Structures, ISPASS 2009*).



For further details, please use the diagram above as a reference. Additional information may be found in this paper http://www.ece.uah.edu/~milenka/docs/vuam_ispass09.pdf that can be downloaded from the class website.

Note that this homework only requires implementation of

- *Branch outcome prediction based on global predictor and bimodal predictor (the blocks in blue labeled as Global predictor and Bimodal table/Bimodal outcome prediction). You can skip loop predictor.*
- *Branch target prediction can be skipped.*

To finish this assignment, open my_predictor.h and implement the member functions of pm_perdicor class.

To compile the predictor, type *make* under ./cbp2-infrastructure-v2/src. You will see a binary program generated, *predict*. To test your local predictor, type *run traces* under ./cbp2-infrastructure-v2/.

Highlights

- *Not required to implement PIR and the hash between IR and PIR.*
- *Please feel free to use any existing C++ code that implements a cache (just cite in your report the code you have used (not required to use or implement C++ class for cache).*
- *Instead of four-way global predictor, you can implement just a two-way global predictor.*

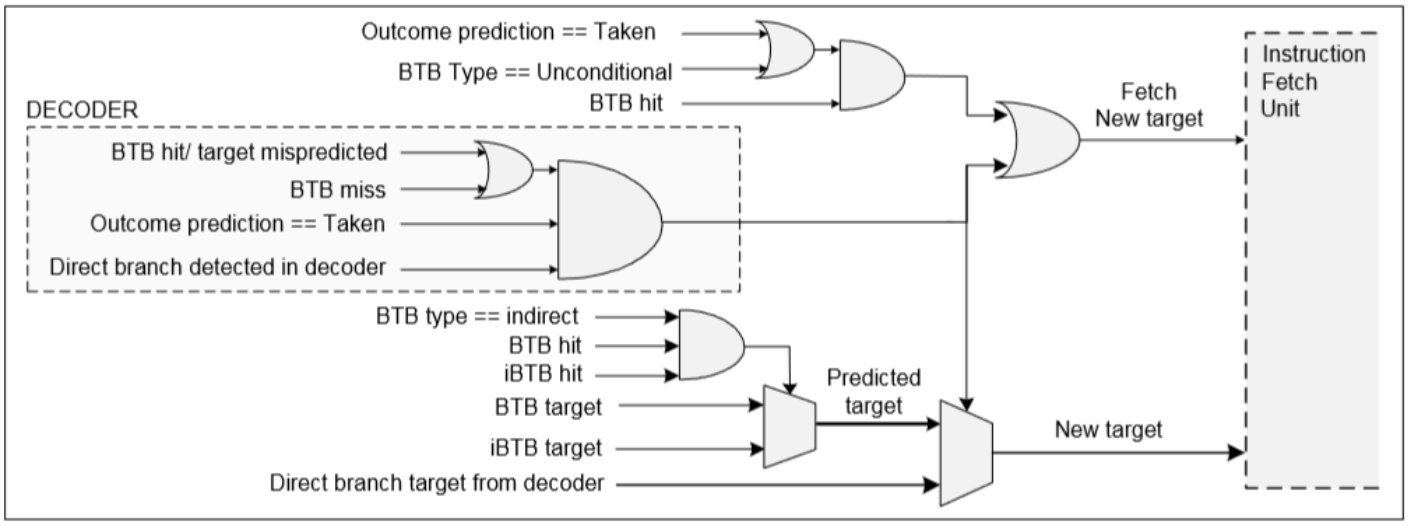
An example of C++ cache implementation is here:
https://github.com/NishadSaraf/L1-Cache-Simulator/

What to Turn In

Please submit your code to github: source code, a report (summarize what you did and results), and a readme file.

Appendix A. The entire Pentium M branch prediction unit.

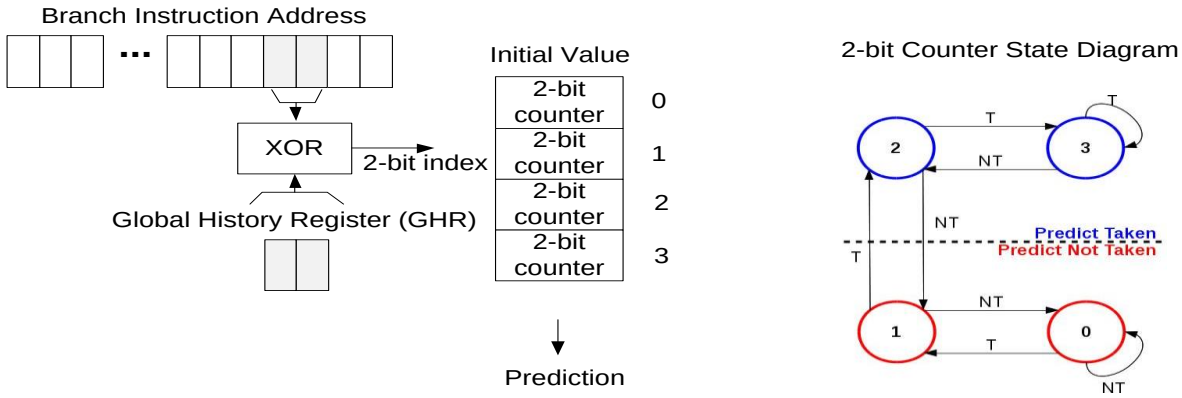
Figure. Complete Pentium M branch prediction units.



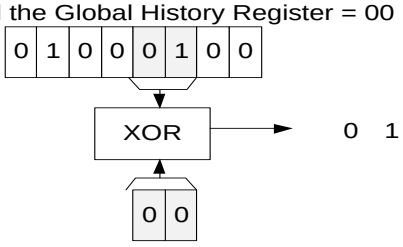
Appendix B. Examples of branch predictors.

The hybrid branch prediction scheme implements two predictors, a local predictor using branch instruction location as index and the second global predictor comprising a four tables of 2 bit counters. You may re-use part of the gshare code for implementing the global predictor. Similar to the design used by Pentium M, when global predictor delivers a hit, the final prediction will be determined by selecting the prediction from the global predictor.

Gshare Branch Predictor



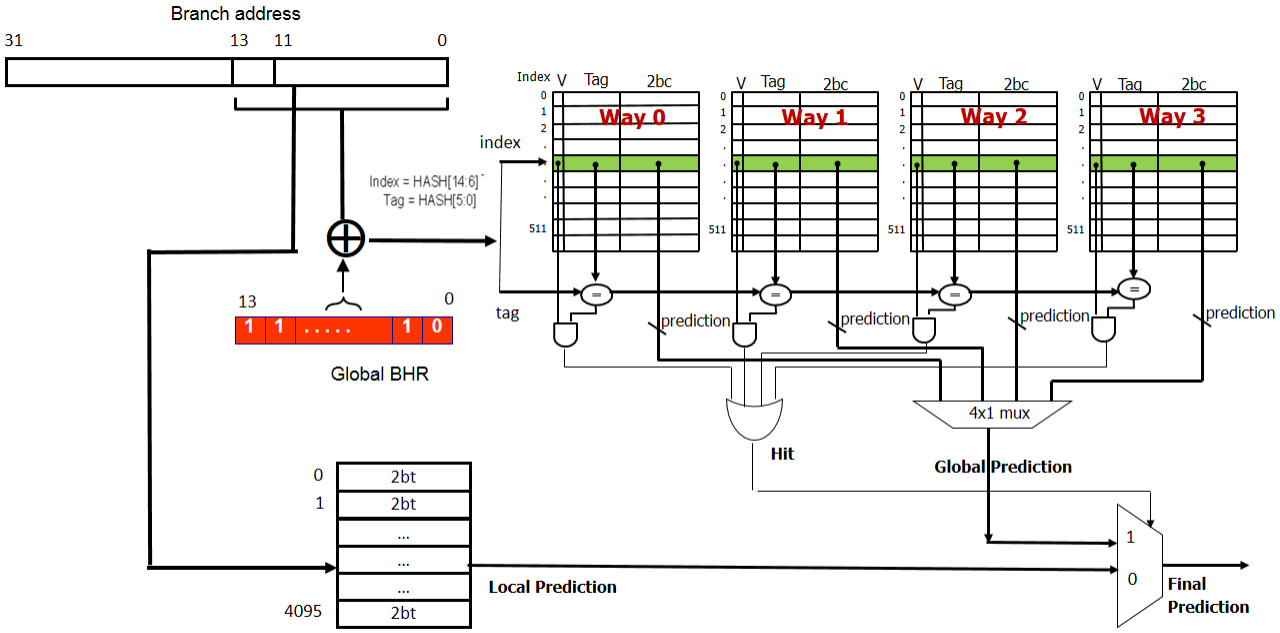
Assume that the hardware is making prediction for b1 and the Global History Register = 00



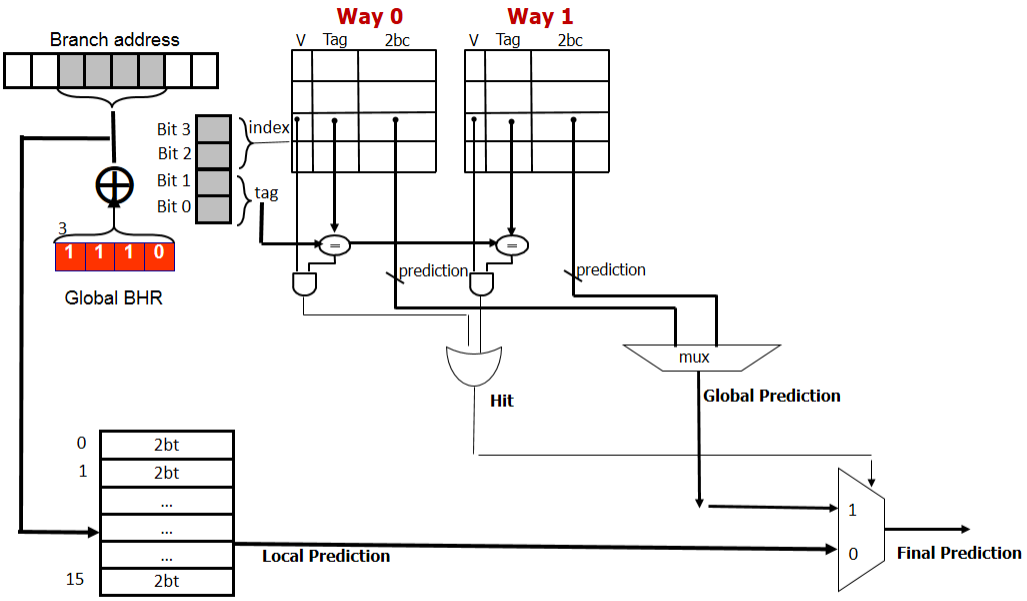
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

To demonstrate how the predictors work, we will use a simplified design and the example program in the second exam. Assume that instruction address has only 8 bits. The local predictor has 16 2 bit counters based on 4-bit index from branch instruction address. The global predictor has 2 tables instead of 4. Each table has 4 2-bit counters and the size of tag is 2.

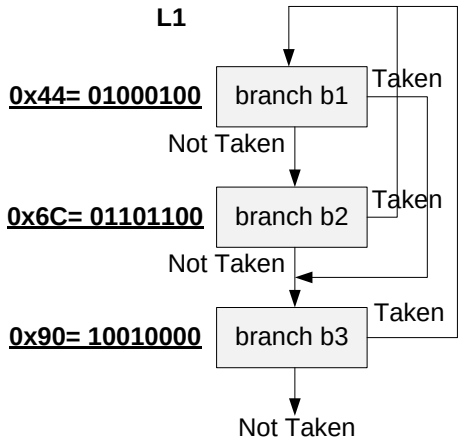
Hybrid Predictor Example



Simplified Hybrid Predictors (only four bits are used for prediction)



Consider the following program (showing only branch instructions and their memory locations),



The table below records branch outcome history for the three branch instructions when the program is executed,

Branch History (1- taken, 0 – not taken)

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**
b1	**b3**	**b1**	**b2**	**b3**	**b1**	**b3**	**b1**	**b2**	**b3**	**b1**	**b2**	**b1**	**b2**	**b3**	**b1**	**b3**	**b1**	**b2**	**b3**	**b1**	**b3**	**b1**
1	1	0	0	1	1	1	0	0	1	0	1	0	0	1	1	1	0	0	1	1	1	0

Table below tracks the value of GHR (4 bits), and branch prediction result.

Hybrid Predictor Results

Col-1		Col-2 Branch Address	Col-3 Branch Actual Outcome	Col-4 GHR	Col-5 4-bit Result (after xor)	Col-6 Global Predictor		Col-7 Predictions		Col-8 Final Prediction
						Row	**Tag**	**Local**	**Global**	
1	**b1**	01**000**100	**1 (T)**	**0000**	**0001**	**0**	**01**	**NT**	**No hit**	**NT**
2	**b3**	10**010**000	**1 (T)**	**0001**	**0101**	**1**	**01**	**NT**	**No hit**	**NT**
3	**b1**	01**000**100	**0 (NT)**	**0011**	**0010**	**0**	**10**	**T**	**No hit**	**T**
4	**b2**	01**101**100	**0 (NT)**	**0110**	**1101**	**3**	**01**	**T**	**No hit**	**T**
5	**b3**	10**010**000	**1 (T)**	**1100**	**1000**	**2**	**00**	**NT**	**No hit**	**NT**
6	**b1**	01**000**100	**1 (T)**	**1001**	**1000**	**2**	**00**	**NT**	**Hit, T**	**T***
7	**b3**	10**010**000	**1 (T)**	**0011**	**0111**	**1**	**11**	**T***	**No hit**	**T***
8	**b1**	01**000**100	**0 (NT)**	**0111**	**0110**	**1**	**10**	**T**	**No hit**	**T**
9	**b2**	01**101**100	**0 (NT)**	**1110**	**0101**	**1**	**01**	**T**	**No hit**	**T**
10	**b3**	10**010**000	**1 (T)**	**1100**	**1000**	**2**	**00**	**T***	**Hit, T**	**T***
11	**b1**	01**000**100	**0 (NT)**	**1001**	**1000**	**2**	**00**	**NT***	**Hit, T**	**T**
12	**b2**	01**101**100	**1 (T)**	**0010**	**1001**	**2**	**01**	**NT**	**No hit**	**NT**
13	**b1**	01**000**100	**0 (NT)**	**0101**	**0100**	**1**	**00**	**NT***	**No hit**	**NT***
14	**b2**	01**101**100	**0 (NT)**	**1010**	**0001**	**0**	**01**	**T**	**Hit, NT**	**NT***
15	**b3**	10**010**000	**1 (T)**	**0100**	**0000**	**0**	**00**	**T***	**No hit**	**T***
16	**b1**	01**000**100	**1 (T)**	**1001**	**1000**	**2**	**00**	**NT**	**Hit, T**	**T***
17	**b3**	10**010**000	**1 (T)**	**0011**	**0111**	**1**	**11**	**T***	**No hit**	**T***
18	**b1**	01**000**100	**0 (NT)**	**0111**	**0110**	**1**	**10**	**NT***	**No hit**	**NT***
19	**b2**	01**101**100	**0 (NT)**	**1110**	**0101**	**1**	**01**	**NT***	**No hit**	**NT***
20	**b3**	10**010**000	**1 (T)**	**1100**	**1000**	**2**	**00**	**T**	**Hit, T**	**T***
21	**b1**	01**000**100	**1 (T)**	**1001**	**1000**	**2**	**00**	**NT**	**Hit, T**	**T***
22	**b3**	10**010**000	**1 (T)**	**0011**	**0111**	**1**	**11**	**T**	**No hit**	**T***

Local Predictor

Col-1			Col-2 4-bit Index	Col-3 16 2-bit Counters																Col-4 Prediction
				0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
1	b1	T	1	0	1*	2	3	0	1	2	3	0	1	2	3	0	1	2	3	NT
2	b3	T	4		2			0*							3					NT
3	b1	NT	1		2*			1							3					T
4	b2	NT	11		1			1							3*					T
5	b3	T	4		1			1*							2					NT
6	b1	T	1		1*			2							2					NT
7	b3	T	4		2			2*							2					T*
8	b1	NT	1		2*			3							2					T
9	b2	NT	11		1			3							2*					T
10	b3	T	4		1			3*							1					T*
11	b1	NT	1		1*			3							1					NT*
12	b2	T	11		0			3							1*					NT
13	b1	NT	1		0*			3							2					NT*
14	b2	NT	11		0			3							2*					T
15	b3	T	4		0			3*							1					T*
16	b1	T	1		0*			3							1					NT
17	b3	T	4		1			3*							1					T*
18	b1	NT	1		1*			3							1					NT*
19	b2	NT	11		0			3							1*					NT*
20	B3	T	4		0			3*							0					T*
21	B1	T	1		0*			3							0					NT
22	B3	T	4		1			3*							0					T*

Global Predictor

Col-1		Col-2		Col-3 4 2-bit Counters (way0)				Col-4 4 2-bit Counters (way1)				Col-5 Hit	Col-6 Prediction
		Row	Tag	valid	Tag	2bc	LRU	Valid	Tag	2bc	LRU		
1	b1 T	0	01	0*	0	0	1	0*	0	3	0	No hit	NT
				0	0	1	1	0	0	2	0		
				0	0	2	1	0	0	1	0		
				0	0	3	1	0	0	0	0		
2	b3 T	1	01	1	01	1	0	0	0	3	1	No hit	NT
				0*	0	1	1	0	0	2	0		
				0	0	2	1	0	0	1	0		
				0	0	3	1	0	0	0	0		
3	b1 N T	0	10	1	01	1	0	0*	0	3	1	No hit	T
				1	01	2	0	0	0	2	1		
				0	0	2	1	0	0	1	0		
				0	0	3	1	0	0	0	0		
4	b2 N T	3	01	1	01	1	1	1	10	2	0	No hit	T
				1	01	2	0	0	0	2	1		
				0	0	2	1	0	0	1	0		
				0*	0	3	1	0	0	0	0		
5	b3 T	2	00	1	01	1	1	1	10	2	0	No hit	T
				1	01	2	0	0	0	2	1		
				0*	0	2	1	0	0	1	0		
				1	01	2	0	0	0	0	1		
6	b1 T	2	00	1	01	1	1	1	10	2	0	Hit	T*
				1	01	2	0	0	0	2	1		
				1*	00	3	0	0	0	1	1		
				1	01	2	0	0	0	0	1		
7	b3 T	1	11	1	01	1	1	1	10	2	0	No hit	T
				1	01	2	0	0*	0	2	1		
				1	00	3	0	0	0	1	1		
				1	01	2	0	0	0	0	1		
8	b1 N T	1	10	1	01	1	1	1	10	2	0	No hit	T
				1*	01	2	1	1	11	3	0		
				1	00	3	0	0	0	1	1		
				1	01	2	0	0	0	0	1		
9	b2 N T	1	01	1	01	1	1	1	10	2	0	No hit	T
				1	10	1	0	1*	11	3	1		
				1	00	3	0	0	0	1	1		
				1	01	2	0	0	0	0	1		
10	b3 T	2	00	1	01	1	1	1	10	2	0	Hit	T*
				1	10	1	1	1	01	2	0		
				1*	00	3	0	0	0	1	1		
				1	01	2	0	0	0	0	1		
11	b1 N T	2	00	1	01	1	1	1	10	2	0	Hit	T
				1	10	1	1	1	01	2	0		
				1*	00	3	0	0	0	1	1		
				1	01	2	0	0	0	0	1		

Col-1		Col-2		Col-3 4 2-bit Counters (way0)				Col-4 4 2-bit Counters (way1)				Col-5 Hit	Col-6 Prediction
		Row	**Tag**	**valid**	**Tag**	**2bc**	**LRU**	**Valid**	**Tag**	**2bc**	**LRU**		
12	**b2 T**	2	01	**1**	**01**	**1**	**1**	**1**	**10**	**2**	**0**	**No hit**	**NT**
				1	**10**	**1**	**1**	**1**	**01**	**2**	**0**		
				1	**00**	**2**	**0**	**0***	**0**	**1**	**1**		
				1	**01**	**2**	**0**	**0**	**0**	**0**	**1**		
13	**b1 NT**	1	00	**1**	**01**	**1**	**1**	**1**	**10**	**2**	**0**	**No hit**	**NT**
				1*	**10**	**1**	**1**	**1**	**01**	**2**	**0**		
				1	**00**	**2**	**1**	**1**	**01**	**2**	**0**		
				1	**01**	**2**	**0**	**0**	**0**	**0**	**1**		
14	**b2 NT**	0	01	**1***	**01**	**1**	**1**	**1**	**10**	**2**	**0**	**Hit**	**NT***
				1	**00**	**0**	**0**	**1**	**01**	**2**	**1**		
				1	**00**	**2**	**1**	**1**	**01**	**2**	**0**		
				1	**01**	**2**	**0**	**0**	**0**	**0**	**1**		
15	**b3 T**	0	00	**1**	**01**	**0**	**0**	**1***	**10**	**2**	**1**	**No hit**	**T**
				1	**00**	**0**	**0**	**1**	**01**	**2**	**1**		
				1	**00**	**2**	**1**	**1**	**01**	**2**	**0**		
				1	**01**	**2**	**0**	**0**	**0**	**0**	**1**		
16	**b1 T**	2	00	**1**	**01**	**0**	**1**	**1**	**00**	**3**	**0**	**Hit**	**T***
				1	**00**	**0**	**0**	**1**	**01**	**2**	**1**		
				1*	**00**	**2**	**1**	**1**	**01**	**2**	**0**		
				1	**01**	**2**	**0**	**0**	**0**	**0**	**1**		
17	**B3 T**	1	11	**1**	**01**	**0**	**1**	**1**	**00**	**3**	**0**	**No hit**	**T**
				1	**00**	**0**	**0**	**1***	**01**	**2**	**1**		
				1	**00**	**3**	**0**	**1**	**01**	**2**	**1**		
				1	**01**	**2**	**0**	**0**	**0**	**0**	**1**		
18	**B1 NT**	1	10	**1**	**01**	**0**	**1**	**1**	**00**	**3**	**0**	**No hit**	**NT**
				1*	**00**	**0**	**1**	**1**	**11**	**3**	**0**		
				1	**00**	**3**	**0**	**1**	**01**	**2**	**1**		
				1	**01**	**2**	**0**	**0**	**0**	**0**	**1**		
19	**B2 NT**	1	01	**1**	**01**	**0**	**1**	**1**	**00**	**3**	**0**	**No hit**	**T**
				1	**10**	**0**	**0**	**1***	**11**	**3**	**1**		
				1	**00**	**3**	**0**	**1**	**01**	**2**	**1**		
				1	**01**	**2**	**0**	**0**	**0**	**0**	**1**		
20	**B3 T**	2	00	**1**	**01**	**0**	**1**	**1**	**00**	**3**	**0**	**Hit**	**T***
				1	**10**	**0**	**1**	**1**	**01**	**2**	**0**		
				1*	**00**	**3**	**0**	**1**	**01**	**2**	**1**		
				1	**01**	**2**	**0**	**0**	**0**	**0**	**1**		
21	**B1 T**	2	00	**1**	**01**	**0**	**1**	**1**	**00**	**3**	**0**	**Hit**	**T***
				1	**10**	**0**	**1**	**1**	**01**	**2**	**0**		
				1*	**00**	**3**	**0**	**1**	**01**	**2**	**1**		
				1	**01**	**2**	**0**	**0**	**0**	**0**	**1**		
22	**B3 T**	1	11	**1**	**01**	**0**	**1**	**1**	**00**	**3**	**0**	**No hit**	**NT**
				1*	**10**	**0**	**1**	**1**	**01**	**2**	**0**		
				1	**00**	**3**	**0**	**1**	**01**	**2**	**1**		
				1	**01**	**2**	**0**	**0**	**0**	**0**	**1**		