

# ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

# ΣΧΟΛΗ ΗΜ&ΜΥ

Προηγμένα Θέματα Αρχιτεκτονικής Υπολογιστών

> 2<sup>η</sup> Άσκηση Ακ. έτος 2011-2012

> > Γρηγόρης Λύρας Α.Μ.: 03109687

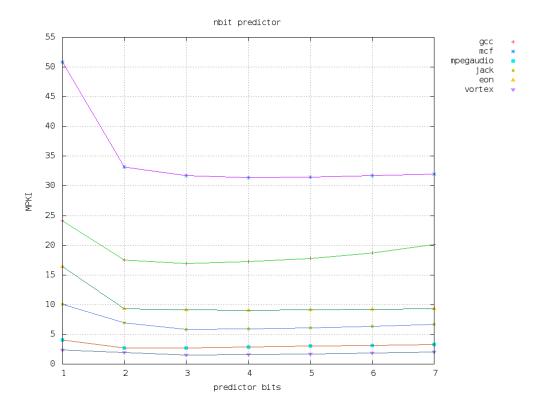
# Εισαγωγή

Στην άσκηση αυτή χρησιμοποιήσαμε ένα Framework για την προσομοίωση αλμάτων όπως έχουν καταγραφεί από την εκτέλεση benchmarks από τη σουίτα SPEC2000. Τα traces περιέχουν μόνο εντολές άλματος όπως αυτές πραγματοποιήθηκαν κατά την εκτέλεση 100M εντολών.

## Μελέτη των n-bit predictors

### **A.1**

Σε αυτό το τμήμα μελετήσαμε την απόδοση των  $\{1..7\}$ -bit predictors χρησιμοποιώντας ως μετρική τα MPKI (Mispredictions Per Thousand Instructions) με 16K BHT entries.



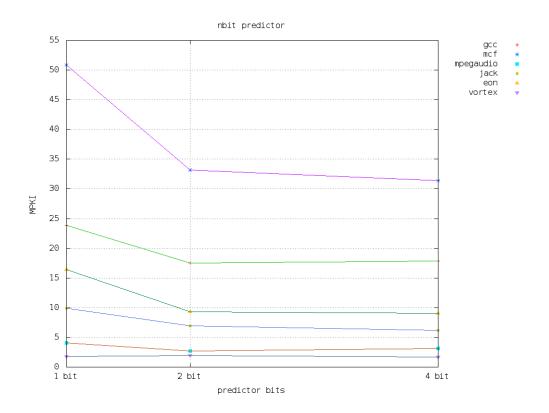
Σχήμα 1: 1-7 bit predictors

Όπως παρατηρούμε από το Σχήμα 1, βέλτιστος προβλέπτης για τα περισσότερα benchmarks είναι ο 4-bit predictor καθώς εκείνος έχει το χαμηλότερο missprediction rate χωρίς να έχει πολύ μεγάλh πολυπλοκότητα στο hardware.

## **A2**

Διαφοροποιούμε την υλοποίηση και μελετάμε {1,2,4}-bit predictors χρησιμοποιώντας ως μετρική τα MPKI (Mispredictions Per Thousand Instructions) με μεταβλητό πλήθος BHT entries ώστε να έχουμε σταθερό μέγεθος hardware και ίσο με 32K.

HW	bits	BHT entries
32K	1	32K
32K	2	16K
32K	4	8K

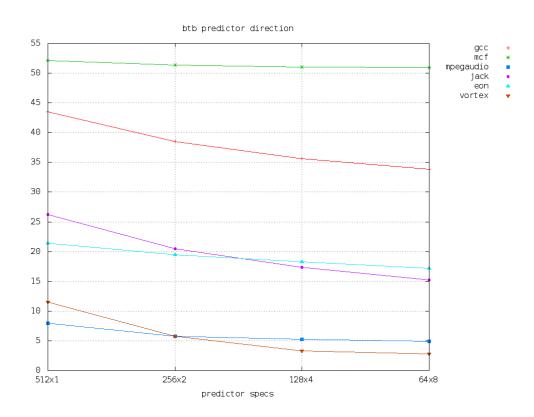


Σχήμα 2: 1,2 και 4 bit predictors

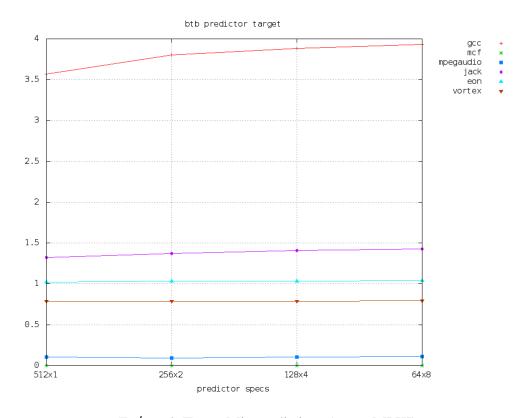
Και σε αυτή την περίπτωση που το hardware είναι περισσότερο (32K) καλύτερα φαίνεται να συμπεριφέρεται ο 4 bit predictor, συνεπώς αυτόν θα επιλέγαμε και σε αυτή τη φορά.

# Β1. Μελέτη του ΒΤΒ

btb_lines	btb_assoc
512K	1
256K	2
128K	4
64K	8



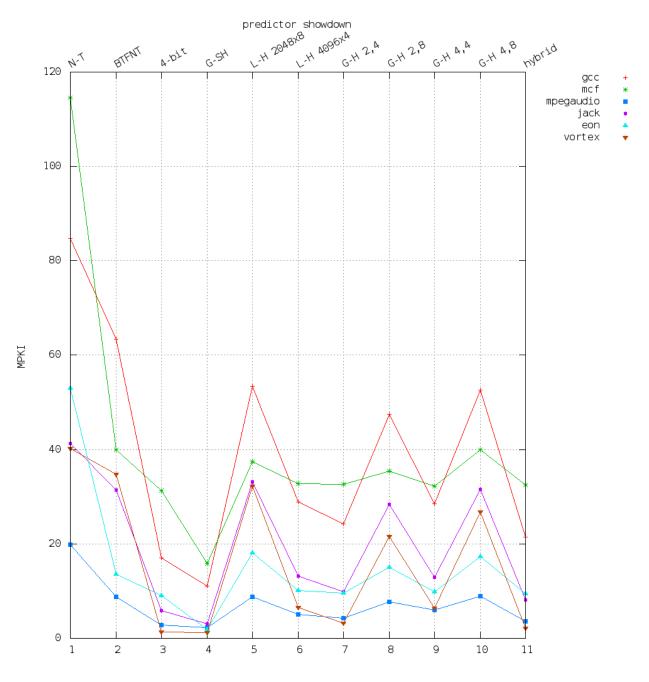
Σχήμα 3: Direction misspredictions (direction MPKI)



Σχήμα 4: Target Misspredictions (target MPKI)

Παρατηρούμε πως το target missprediction παραμένει σταθερό σχεδόν, σε αντίθεση με το direction missprediction που μειώνεται δραστικά στον 64x8 btb predictor. Αυτή είναι και η επιθυμητή οργάνωση για τον btb.

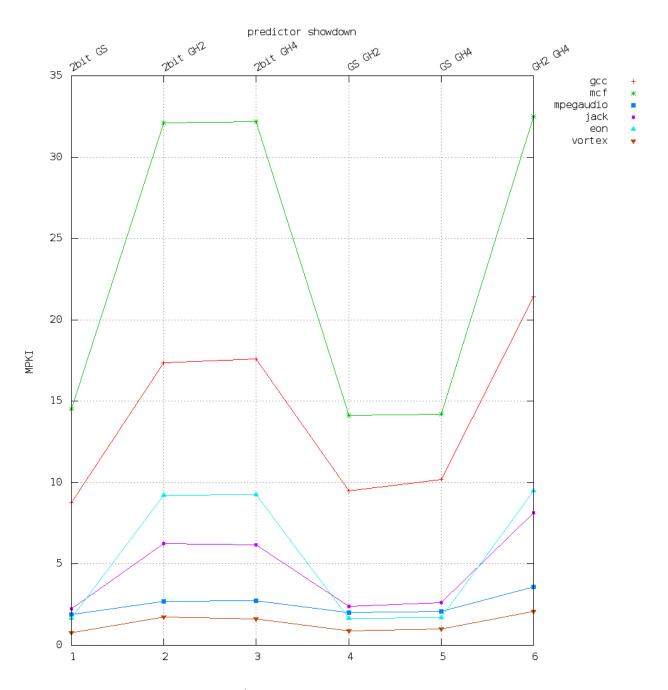
# C1. Σύγκριση διαφορετικών predictors



Σχήμα 5: Σύγκριση predictors

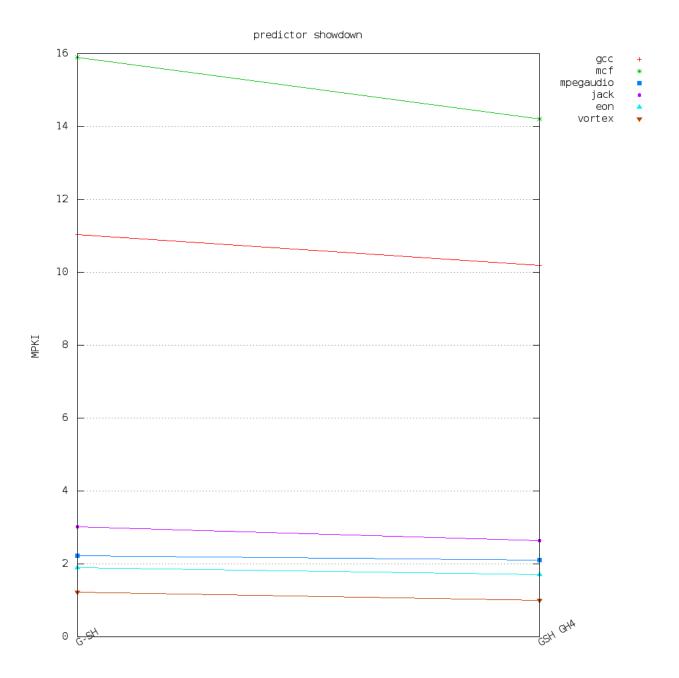
Από αυτούς καταλήγουμε πως καλύτερος είναι ο gshare predictor.

## **Tournament Hybrid predictors**



Σχήμα 6: Σύγκριση hybrid predictors

Από όλους τους hybrid predictors που χρησιμοποιήσαμε, καλύτεροι αποδείχτηκαν οι gshare-globalhistory2 και gshare-globalhistory4 predictors. Τέλος συγκρίνουμε τον hybrid gshare-globalhistory4 και gshare predictor.



Σχήμα 7: Σύγκριση hybrid predictors

Καταλήγουμε πως ο hybrid predictor που δημιουργήσαμε είναι ελαρφώς καλύτερος από τον gshare predictor.

Ο πηγαίος κώδικας που χρησιμοποιήσαμε ήταν ο ακόλουθος:

```
// This file contains the main function. The program accepts a single // parameter: the name of a trace file. It drives the branch predictor
2
3
      // simulation by reading the trace file and feeding the traces one at a time
      // to the branch predictor.
5
 6
      #include <stdio.h>
      #include <stdlib.h>
      #include <string.h>
      #include <assert.h>
10
12
      #include <algorithm>
13
14
      #include "branch.h"
15
```

```
#include "trace.h"
16
     #include "predictor.h"
17
     #include "btb.h"
18
     {\it \#include \ "nt\_predictor.h"}
19
20
     #include "btfnt_predictor.h"
     #include "nbit_predictor.h"
                                                  //the .h files of the branch predictors' implementations
21
     #include "gshare_predictor.h"
22
     #include "localhistory_predictor.h"
23
     #include "globalhistory_predictor.h"
24
     #include "hybrid_predictor.h"
25
26
27
     #define NR_PREDICTORS 11
28
29
30
     using namespace std;
31
     int main (int argc, char *argv[])
32
33
34
35
         // make sure there is one parameter
36
         if (argc != 2) {
37
38
             fprintf (stderr, "Usage: %s <filename>.gz\n", argv[0]);
39
             exit (1);
40
41
42
         // open the trace file for reading
43
44
         init_trace (argv[1]);
45
46
         // initialize competitor's branch prediction code
47
48
         // you can use more than one predictor in an array of predictors!!!
49
         branch_predictor **p = new branch_predictor*[NR_PREDICTORS];
50
51
52
         p[0] = new nt_predictor();
         p[1] = new btfnt_predictor();
53
54
         p[2] = new nbit_predictor(4);
55
         p[3] = new gshare_predictor();
         p[4] = new localhistory_predictor(2048, 8);
56
57
         p[5] = new localhistory_predictor(4096, 4);
             /* X=2 BHR=4 */
58
         p[6] = new globalhistory_predictor(16384, 2, 4);
59
             /* X=2 BHR=8 */
60
         p[7] = new globalhistory_predictor(16384, 2, 8);
61
62
             /* X=4 BHR=4 */
         p[8] = new globalhistory_predictor(8192, 4, 4);
63
64
              /* X=4 BHR=8 */
65
         p[9] = new globalhistory_predictor(8192, 4, 8);
66
67
         p[10] = new hybrid_predictor(512);
68
         long long int
69
                                       // number of target mispredictions
70
         tmiss[NR_PREDICTORS],
71
               dmiss[NR_PREDICTORS];
                                               // number of direction mispredictions
72
         fill( tmiss, tmiss+NR_PREDICTORS, 0);
73
74
         fill( dmiss, dmiss+NR_PREDICTORS, 0);
75
         // keep looping until end of file
76
         for (;;) {
77
78
             // get a trace
79
80
81
             trace *t = read_trace ();
82
             // NULL means end of file
83
84
             if (!t) break;
85
86
87
             // send this trace to the competitor's code for prediction
88
             branch_update *u;
89
90
```

```
91
                       /* static not taken */
              u = p[0] - predict(t->bi);
92
              p[0]->update(u, t->taken, t->target);
93
              dmiss[0] += u->direction_prediction() != t->taken;
94
95
              tmiss[0] += u->target_prediction() != t->target;
96
97
98
                       /* backward taken forward not taken */
99
              ((btfnt_predictor *)p[1])->set_target(t->target > t->bi.address);
100
              u = p[1] - predict(t->bi);
              p[1]->update(u, t->taken, t->target);
101
102
              dmiss[1] += u->direction_prediction() != t->taken;
              tmiss[1] += u->target_prediction() != t->target;
103
104
105
                       /* 4 bit predictor */
106
              u = p[2] - predict(t->bi);
107
108
              p[2]->update(u, t->taken, t->target);
              dmiss[2] += u->direction_prediction() != t->taken;
109
110
              tmiss[2] += u->target_prediction() != t->target;
111
                       /* gshare predictor */
112
              u = p[3] - predict(t->bi);
113
              p[3]->update(u, t->taken, t->target);
114
115
              dmiss[3] += u->direction_prediction() != t->taken;
              tmiss[3] += u->target_prediction() != t->target;
116
117
118
                       /* local history 2 level predictor X=2048 */
              u = p[4] - predict(t->bi);
119
              p[4]->update(u, t->taken, t->target);
120
121
              dmiss[4] += u->direction_prediction() != t->taken;
              tmiss[4] += u->target_prediction() != t->target;
122
123
                       /* local history 2 level predictor X=4096 */
124
              u = p[5] - predict(t->bi);
125
              p[5]->update(u, t->taken, t->target);
126
127
              dmiss[5] += u->direction_prediction() != t->taken;
              tmiss[5] += u->target_prediction() != t->target;
128
129
                       /* global history X=2 BHR=4 */
130
              u = p[6]->predict(t->bi);
131
132
              p[6]->update(u, t->taken, t->target);
              dmiss[6] += u->direction_prediction() != t->taken;
133
134
              tmiss[6] += u->target_prediction() != t->target;
135
                       /* global history X=2 BHR=8 */
136
137
              u = p[7] - predict(t->bi);
              p[7]->update(u, t->taken, t->target);
138
              dmiss[7] += u->direction_prediction() != t->taken;
139
140
              tmiss[7] += u->target_prediction() != t->target;
141
142
                       /* global history X=4 BHR=4 */
              u = p[8]->predict(t->bi);
143
              p[8]->update(u, t->taken, t->target);
144
145
              dmiss[8] += u->direction_prediction() != t->taken;
              tmiss[8] += u->target_prediction() != t->target;
146
147
                       /* global history X=4 BHR=8 */
148
              u = p[9] - predict(t->bi);
149
150
              p[9]->update(u, t->taken, t->target);
              dmiss[9] += u->direction_prediction() != t->taken;
151
              tmiss[9] += u->target_prediction() != t->target;
152
153
              u = p[10] - predict(t->bi);
154
155
              p[10]->update(u, t->taken, t->target);
              dmiss[10] += u->direction_prediction() != t->taken;
156
              tmiss[10] += u->target_prediction() != t->target;
157
158
159
          // done reading traces
160
161
162
          end_trace ();
163
          // give final mispredictions per kilo-instruction and exit.
164
          // each trace represents exactly 100 million instructions.
165
```

#### **Static Not-Taken**

```
1
     /* -.-.-.-.-.-.
2
     * File Name : nt\_predictor.h
     * Creation Date : 19-05-2012
3
     * Last Modified : Sat 19 May 2012 10:09:02 PM EEST
4
     * Created By : Greg Liras <gregliras@gmail.com>
5
     6
     // nt predictor.h
8
9
10
     \#ifndef\ NT\_PREDICTOR\_H
     #define NT_PREDICTOR_H
#include "predictor.h"
11
12
13
14
15
     class nt_update : public branch_update
16
    public:
17
18
        unsigned int index;
19
20
     class nt_predictor: public branch_predictor
21
        nt_update u:
22
23
        branch_update *predict (branch_info & b);
        void update (branch_update *u, bool taken, unsigned int target);
24
    };
25
26
27
     #endif
1
     /* -.-.-.-.-.-.
     * File Name : nt\_predictor.cpp
2
     * Creation Date : 19-05-2012
3
     * Last Modified : Sat 19 May 2012 10:07:46 PM EEST
4
5
     * Created By : Greg Liras <gregliras@gmail.com>
6
      _----*/
     \#include "nt\_predictor.h"
10
     branch_update *nt_predictor::predict (branch_info & b)
11
        if (b.br flags & BR CONDITIONAL) {
12
13
            u.direction_prediction (false);
        } else {
14
15
            u.direction_prediction (true);
16
17
        u.target_prediction (0);
18
        return &u;
20
     void nt_predictor::update (branch_update *u, bool taken, unsigned int target)
21
    }
```

#### Static Backward Taken Forward Not Taken

```
8
9
     #ifndef BTFNT_PREDICTOR_H
     #define BTFNT_PREDICTOR_H
10
11
12
     #include <math.h>
     #include "predictor.h"
13
15
     class btfnt_predictor : public branch_predictor
16
     public:
17
18
19
         branch_update u;
         branch_info bi;
20
         bool jump;
21
22
             void set_target(bool t);
23
         branch_update *predict (branch_info & b);
24
25
         void update (branch_update *u, bool taken, unsigned int target);
     };
26
27
28
     #endif
     /* -.-.-.-.-.-.
1
      * File Name : btfnt_predictor.cpp
2
      * Creation Date : 19-05-2012
      * Last Modified : Sat 26 May 2012 06:03:49 PM EEST
4
      * Created By : Greg Liras \langle gregliras@gmail.com \rangle
5
     #include "btfnt_predictor.h"
6
7
8
     void btfnt_predictor::set_target (bool t) {
             jump = t;
10
11
     branch_update *btfnt_predictor::predict (branch_info & b)
12
13
14
         if (b.br_flags & BR_CONDITIONAL) {
15
16
             if (jump) {
17
                             u.direction_prediction (false);
18
19
             else {
                             u.direction_prediction(true);
20
                     }
21
         } else {
23
             {\tt u.direction\_prediction\ (true);}
24
25
26
         u.target_prediction (0);
27
28
     void btfnt_predictor::update (branch_update *u, bool taken, unsigned int target)
29
30
31
    }
32
```

## 4-bit predictor

```
// nbit_predictor.h
2
     //
3
4
     #ifndef N_BIT_PREDICTOR_H
5
     {\it \#define N\_BIT\_PREDICTOR\_H}
8
     #include <math.h>
     #include <string.h>
10
     #include "predictor.h"
11
12
     class nbit_update : public branch_update
13
14
     public:
15
         unsigned int index;
```

```
|};
17
18
     class nbit_predictor : public branch_predictor
19
20
     public:
21
     #define NBP_TABLE_BITS
                                   \frac{15}{\sqrt{number of entries}} = 2^15
22
         nbit_update u;
23
         branch_info bi;
24
25
         int counter limit:
         int N_COUNTER_LENGTH;
26
27
         unsigned char tab[1<<NBP_TABLE_BITS];</pre>
28
29
         nbit_predictor (int length);
30
31
         branch_update *predict (branch_info & b);
         void update (branch_update *u, bool taken, unsigned int target);
32
     };
33
34
    #endif
35
1
2
      * File Name : nbit_predictor.cpp
      * Creation Date : 15-05-2012
3
      * Last Modified : Tue 15 May 2012 05:25:25 PM EEST
4
      * Created By : Greg Liras <gregliras@gmail.com>
     #include "nbit_predictor.h"
6
7
8
     nbit_predictor::nbit_predictor (int length) :N_COUNTER_LENGTH(length)
9
10
11
         memset (tab, 0, sizeof (tab));
         counter_limit = ((int) pow(2.0, N_COUNTER_LENGTH)) - 1;
12
13
     }
14
15
     branch_update *nbit_predictor::predict (branch_info & b)
16
17
         if (b.br_flags & BR_CONDITIONAL) {
18
19
             u.index = (b.address & ((1<<NBP_TABLE_BITS)-1));</pre>
             u.direction_prediction (tab[u.index] >> (N_COUNTER_LENGTH-1));
20
21
         } else {
22
             u.direction_prediction (true);
23
         u.target_prediction (0);
25
         return &u:
     7
26
27
     void nbit_predictor::update (branch_update *u, bool taken, unsigned int target)
28
29
         if (bi.br_flags & BR_CONDITIONAL) {
30
             unsigned char *c = &tab[((nbit_update*)u)->index];
31
32
             if (taken) {
                 if (*c < counter_limit) (*c)++;</pre>
33
34
             } else {
                 if (*c > 0) (*c)--;
35
36
37
         }
```

## gshare predictor

```
// gshare_predictor.h
1
2
     // This file contains a sample my_predictor class.
     // It is a simple 32,768-entry gshare with a history length of 15.
3
4
     \#ifndef\ GSHARE\_PREDICTOR\_H
     #define GSHARE_PREDICTOR_H
6
     #include "predictor.h"
9
10
     class gshare_update : public branch_update
11
     public:
12
```

```
unsigned int index;
13
     };
14
15
16
17
     * H klash gshare_predictor klhronomei thn klash
     * branch_predictor kai kanei override tis me8odous
18
      * predict kai update
19
20
21
     class gshare_predictor : public branch_predictor
22
23
24
     public:
     #define HISTORY_LENGTH
25
                                   14
     \#define\ GSP\_TABLE\_BITS
26
27
         gshare_update u;
         branch info bi;
28
29
         unsigned int history;
30
         unsigned char tab[1<<GSP_TABLE_BITS];</pre>
31
32
         gshare_predictor (void);
33
         branch_update *predict (branch_info & b);
34
35
         void update (branch_update *u, bool taken, unsigned int target);
     };
36
37
     #endif
     1
     * File Name : gshare_predictor.cpp
2
     * Creation Date : 20-05-2012
3
     * Last Modified : Sat 26 May 2012 06:16:06 PM EEST
4
     * Created By : Greg Liras <gregliras@gmail.com>
5
6
     _._...*/
     #include "qshare predictor.h"
8
9
     #include <algorithm>
     gshare_predictor::gshare_predictor (void) : history(0)
10
11
12
         std::fill (tab, tab+sizeof (tab),0);
13
     branch_update *gshare_predictor::predict (branch_info & b)
14
15
         bi = b;
16
17
18
         * O qshare xrhsimopoieitai mono qia conditional branches.
19
          st Ta uncoditional ginontai predicted panta TAKEN
20
21
22
         if (b.br_flags & BR_CONDITIONAL) {
23
             u.index = (history << (GSP_TABLE_BITS - HISTORY_LENGTH)) ^ (b.address & ((1<<GSP_TABLE_BITS)-1));
24
             u.direction_prediction (tab[u.index] >> 1);
25
26
         } else {
             u.direction_prediction (true);
27
28
29
         // O gshare den kanei target prediction, gia auto to 8etoume sto O.
30
31
32
         u.target_prediction (0);
         return &u;
33
34
     void gshare_predictor::update (branch_update *u, bool taken, unsigned int target)
35
36
         //O gshare xrhsimopoieitai mono gia conditional branches
37
         if (bi.br_flags & BR_CONDITIONAL) {
38
39
             unsigned char *c = &tab[((gshare_update*)u)->index];
40
             if (taken) {
41
                 if (*c < 3) (*c)++;
42
             } else {
                 if (*c > 0) (*c)--;
43
44
45
             history <<= 1;
             history |= taken;
46
             history &= (1<<HISTORY_LENGTH)-1;</pre>
47
48
    }
49
```

## **Local-History two-level predictors**

```
\#ifndef\ local history\_predictor\_H
1
2
     {\it \#define\ localhistory\_predictor\_H}
3
     #include <cmath>
4
     \#include \ "predictor.h"
6
     class localhistory_update : public branch_update
     public:
9
10
         unsigned int phtindex;
         unsigned int bhtindex;
11
12
13
     class localhistory_predictor : public branch_predictor
14
15
16
     public:
17
18
         localhistory_update u;
         branch_info bi;
19
20
21
              int p_counter_limit;
22
              int b_counter_limit;
23
24
              int pht_entries;
              int pht_nbit;
25
26
              int bht_entries;
              int bht_entry_length;
28
29
         unsigned char *pht;
30
         unsigned char *bht;
31
32
              unsigned int pht_mask;
33
34
35
         localhistory_predictor (int x,int z);
36
37
         branch_update *predict (branch_info & b);
38
          void update (branch_update *u, bool taken, unsigned int target);
39
40
     };
41
     #endif
     #include "localhistory_predictor.h"
1
     #include <algorithm>
2
     #include <cstring>
4
     localhistory_predictor::localhistory_predictor (int x , int z) : bht_entries(x), bht_entry_length(z)
5
     {
6
              pht_entries=8192;
              pht_nbit=2;
9
10
              pht = new unsigned char [pht_entries];
              memset (pht, 0, sizeof (pht));
11
12
              bht = new unsigned char [x];
13
              memset (bht, 0, sizeof (bht));
14
15
16
              pht_mask = ((1<<(((int) log2(pht_entries))-bht_entry_length))-1);</pre>
17
              p_counter_limit = (1<<pht_nbit);</pre>
18
              b_counter_limit = (1<<bht_entry_length);</pre>
20
21
22
     branch_update *localhistory_predictor::predict (branch_info & b)
23
24
              if (b.br_flags & BR_CONDITIONAL) {
25
26
27
                      u.bhtindex = (b.address & (bht_entries-1));
                      u.phtindex = ((b.address & pht_mask)<<bht_entry_length);</pre>
28
                      u.phtindex |= bht[u.bhtindex];
29
30
                      u.direction_prediction(pht[u.phtindex]>>(pht_nbit-1));
31
```

```
} else {
32
33
                      u.direction_prediction(true);
34
35
              u.target_prediction (0);
36
              return &u;
     }
37
38
     void localhistory_predictor::update (branch_update *u, bool taken, unsigned int target)
39
40
         if (bi.br_flags & BR_CONDITIONAL) {
41
42
43
                      unsigned char *c = &pht[((localhistory_update*)u)->phtindex];
                      unsigned char *d = &bht[((localhistory_update*)u)->bhtindex];
44
45
46
                      if (taken) {
                               if (*c < p_counter_limit)</pre>
47
48
                                        (*c)++;
49
                      }
                      else {
50
                               if (*c > 0)
51
52
53
54
                       (*d) <<= 1;
55
                       (*d) |= taken;
                       (*d) &= b_counter_limit;
56
57
58
59
              }
    }
60
```

## Global-History two-level predictors

```
#ifndef GLOBALHISTORY_PREDICTOR_H
1
     #define GLOBALHISTORY_PREDICTOR_H
2
     #include <cmath>
4
     #include <cstring>
     #include <cstdio>
6
     #include "predictor.h"
     class globalhistory_update : public branch_update
10
11
     public:
         unsigned int index;
12
13
     };
14
     {\tt class~globalhistory\_predictor}~:~{\tt public~branch\_predictor}
15
16
     public:
17
18
19
          globalhistory_update u;
20
         branch_info bi;
21
              int p_counter_limit;
22
              int b_counter_limit;
23
24
              unsigned int pht_mask;
25
              int pht_entries;
26
27
              int pht_nbit;
28
29
              int bhr;
30
              int bhr_length;
31
         unsigned char **pht;
32
33
          globalhistory_predictor (int x,int y, int z);
34
35
          branch_update *predict (branch_info & b);
36
37
          void update (branch_update *u, bool taken, unsigned int target);
     };
38
     #endif
39
    #include "globalhistory_predictor.h"
    #include <algorithm>
2
```

```
3
     globalhistory_predictor::globalhistory_predictor (int x , int y, int z) :pht_entries(x), pht_nbit(y), bhr_length(z)
4
5
7
              bhr=0;
8
              pht = new unsigned char *[1<<bhr_length];</pre>
10
              for( int i = 0; i < (1<<bhr_length); i++)</pre>
11
12
              {
                      pht[i] = new unsigned char [pht_entries>>bhr_length];
13
14
15
              for (int i = 0; i < (1<<bhr_length); i++)</pre>
16
17
                      memset(pht[i], 0, sizeof (pht[i]));
18
              }
19
20
              p_counter_limit = (1<<pht_nbit);</pre>
21
22
              b_counter_limit = (1<<bhr_length);</pre>
23
              pht_mask = ((pht_entries>>bhr_length)-1);
     }
24
25
     branch_update *globalhistory_predictor::predict (branch_info & b)
26
27
28
             if (b.br_flags & BR_CONDITIONAL) {
29
30
                      u.index = (b.address & pht_mask);
31
32
                      u.direction_prediction(pht[bhr][u.index]>>(pht_nbit-1));
33
              } else {
34
35
                      u.direction_prediction (true);
36
              u.target_prediction (0);
37
38
              return &u;
39
40
41
     void globalhistory_predictor::update (branch_update *u, bool taken, unsigned int target)
42
         if (bi.br_flags & BR_CONDITIONAL) {
43
44
                      unsigned char *c = &pht[bhr][((globalhistory_update*)u)->index];
45
46
47
                      if (taken) {
                              if (*c < p_counter_limit)</pre>
48
49
                                       (*c)++;
50
51
                      else {
52
                              if (*c > 0)
                                       (*c)--;
53
54
                      }
                      bhr <<= 1;
55
                      bhr |= taken:
56
57
                      bhr &= b_counter_limit;
58
              }
59
    }
     * File Name : hybrid_predictor.h
2
     * Creation Date : 27-05-2012
3
     * Last Modified : Sun 27 May 2012 11:04:25 PM EEST
     * Created By : Greg Liras <gregliras@gmail.com>
5
     _._....*/
     {\it \#ifndef~HYBRID\_PREDICTOR\_H}
8
     #define HYBRID_PREDICTOR_H
9
10
     \#include\ "predictor.h"
11
     #include "globalhistory_predictor.h"
12
13
14
     class hybrid_update : public branch_update
15
     public:
16
```

```
17
             branch_update *ups[2];
             unsigned int index;
18
             unsigned int pred;
19
20
21
22
     class hybrid_predictor : public branch_predictor
23
     public:
24
25
             hybrid_update u;
26
             branch_info bi;
27
28
             int counter_limit;
29
             branch_predictor **preds;
30
31
32
33
             int pht_entries;
34
             int pht_bits_length;
35
36
             unsigned char *tab;
37
             hybrid_predictor(int entries);
38
39
         branch_update *predict (branch_info & b);
40
         void update (branch_update *u, bool taken, unsigned int target);
41
42
43
44
45
     };
46
47
    #endif
48
     /* -.-.-.-.-.-.
1
2
     * File Name : hybrid_predictor.cpp
     * Creation Date : 27-05-2012
3
     * Last Modified : Sun 27 May 2012 11:09:18 PM EEST
4
5
     * Created By : Greg Liras \langle gregliras@gmail.com \rangle
6
     #include "hybrid_predictor.h"
9
     hybrid_predictor::hybrid_predictor(int entries) : pht_entries(entries)
10
11
             tab = new unsigned char[pht_entries];
12
13
             memset(tab,0,sizeof(tab));
             pht_bits_length = 2;
14
             counter_limit = 2;
15
16
             u.pred = 0;
             u.index = 0;
17
18
             preds = new branch_predictor*[2];
19
     }
20
21
22
     branch_update *hybrid_predictor::predict (branch_info & b)
23
24
         bi = b;
         if (b.br_flags & BR_CONDITIONAL) {
25
             u.index = (b.address & (pht_entries-1));
26
             u.pred = tab[u.index] >> (pht_bits_length - 1);
27
28
             u.ups[0] = (branch_update *) preds[0]->predict(b);
29
             u.ups[1] = (branch_update *) preds[1]->predict(b);
30
             unsigned int thepred = u.pred & 1;
31
32
             u.direction_prediction(u.ups[thepred]->direction_prediction());
33
34
     }
35
     void hybrid_predictor::update (branch_update *u, bool taken, unsigned int target)
36
         if (bi.br_flags & BR_CONDITIONAL) {
37
38
             unsigned char *c = &tab[((hybrid_update*)u)->index];
                      bool pred0_result = ((hybrid_update *)u)->ups[0]->direction_prediction() == taken;
39
40
                      bool pred1_result = ((hybrid_update *)u)->ups[1]->direction_prediction() == taken;
41
                      int meta_update = pred1_result - pred0_result;
42
```

```
if (meta\_update > 0 \&\& *c < 3)
43
44
                                       (*c)++;
                      else if (meta_update < 0 && *c > 0)
45
46
                                       (*c)--;
47
                      (*c) &= 3;
                      preds[0]->update(((hybrid_update *)u)->ups[0], taken, target);
48
49
                      preds[1]->update(((hybrid_update *)u)->ups[1], taken, target);
50
51
    }
```

#### 2-bit global history BHT=2

```
1
    * File Name : hybrid_2bit_GH2_predictor.cpp
2
    * Creation Date : 27-05-2012
3
    * Last Modified : Sun 27 May 2012 10:38:21 PM EEST
4
5
    * Created By : Greg Liras <gregliras@gmail.com>
6
    7
8
Q
10
    #include "hybrid_2bit_GH2_predictor.h"
11
12
13
    hybrid_2bit_GH2_predictor::hybrid_2bit_GH2_predictor(int entries): hybrid_predictor(entries)
14
15
            preds[0] = new nbit_predictor(2);
16
            preds[1] = new globalhistory_predictor(8192,2,2);
17
    }
18
    1
2
    * File Name : hybrid_2bit_GH2_predictor.h
    * Creation Date : 27-05-2012
3
    * Last Modified : Sun 27 May 2012 10:45:58 PM EEST
4
    * Created By : Greg Liras <gregliras@gmail.com>
    _-----*/
6
8
9
10
    #ifndef HYBRID_2BIT_GH2_PREDICTOR_H
11
    #define HYBRID_2BIT_GH2_PREDICTOR_H
12
13
    #include "predictor.h"
14
15
    {\it \#include \ "gshare\_predictor.h"}
    #include "nbit_predictor.h"
16
    #include "hybrid_predictor.h"
17
18
19
    class hybrid_2bit_GH2_predictor : public hybrid_predictor
20
21
    public:
22
            hybrid_2bit_GH2_predictor(int entries);
23
24
25
26
    };
27
28
29
    #endif
```

### 2-bit global history BHT=4

```
10
     #include "hybrid_2bit_GH4_predictor.h"
11
12
13
14
     hybrid_2bit_GH4_predictor::hybrid_2bit_GH4_predictor(int entries): hybrid_predictor(entries)
15
             preds[0] = new nbit_predictor(2);
16
             preds[1] = new globalhistory_predictor(8192,2,4);
17
     }
18
1
     /* -.-.-.-.-.
     * File Name : hybrid_2bit_GH4_predictor.h
2
3
     * Creation Date : 27-05-2012
     * Last Modified : Sun 27 May 2012 10:46:01 PM EEST
4
5
     * Created By : Greg Liras <gregliras@gmail.com>
6
     _------*/
9
10
     {\it \#ifndef~HYBRID\_2BIT\_GH4\_PREDICTOR\_H}
11
     #define HYBRID_2BIT_GH4_PREDICTOR_H
12
13
     #include "predictor.h"
14
     #include "gshare_predictor.h"
#include "nbit_predictor.h"
15
16
     #include "hybrid_predictor.h"
17
18
19
     class hybrid_2bit_GH4_predictor : public hybrid_predictor
20
     public:
21
22
             hybrid_2bit_GH4_predictor(int entries);
23
24
25
     };
26
27
28
     #endif
29
```

### 2-bit gshare

```
1
                  -.-.-.-.-.-.
     *\ \textit{File Name : hybrid\_2bit\_GS\_predictor.cpp}
2
3
     * Creation Date : 27-05-2012
     * Last Modified : Sun 27 May 2012 10:34:58 PM EEST
4
5
     * Created By : Greg Liras \langle gregliras@gmail.com \rangle
     6
8
     {\it \#include \ "hybrid\_2bit\_GS\_predictor.h"}
9
10
     hybrid_2bit_GS_predictor::hybrid_2bit_GS_predictor(int entries): hybrid_predictor(entries)
11
12
            preds[0] = new nbit_predictor(2);
13
            preds[1] = new gshare_predictor();
14
    }
15
     * File Name : hybrid_2bit_GS_predictor.h
2
     * Creation Date : 27-05-2012
3
     * Last Modified : Sun 27 May 2012 10:46:04 PM EEST
4
     * Created By : Greg Liras <gregliras@gmail.com>
5
6
     _._._...*/
8
9
     #ifndef HYBRID_2BIT_GS_PREDICTOR_H
10
     #define HYBRID_2BIT_GS_PREDICTOR_H
11
12
     {\it \#include \ "predictor.h"}
13
     #include "gshare_predictor.h"
14
     #include "nbit_predictor.h"
15
     #include "hybrid_predictor.h"
16
```

```
17
     class hybrid_2bit_GS_predictor : public hybrid_predictor
18
19
20
     public:
21
              hybrid_2bit_GS_predictor(int entries);
22
23
24
25
26
     };
27
28
29
     #endif
30
```

## global history BHT=2,4

```
1
     * File Name : hybrid_GH2_GH4_predictor.cpp
2
     * Creation Date : 27-05-2012
     * Last Modified : Sun 27 May 2012 11:16:20 PM EEST
4
5
     * Created By : Greg Liras <gregliras@gmail.com>
     _._._.*/
8
10
11
     #include "hybrid_GH2_GH4_predictor.h"
12
13
14
     hybrid_GH2_GH4_predictor::hybrid_GH2_GH4_predictor(int entries): hybrid_predictor(entries)
15
16
             preds[0] = new globalhistory_predictor(8192,2,2);
17
             preds[1] = new globalhistory_predictor(8192,2,4);
    }
18
     * File Name : hybrid_GH2_GH4_predictor.h
2
     * Creation Date : 27-05-2012
3
     * Last Modified : Sun 27 May 2012 11:16:25 PM EEST
     * Created By : Greg Liras <gregliras@gmail.com>
5
     _._....*/
8
9
10
11
     #ifndef HYBRID_GH2_GH4_PREDICTOR_H
     #define HYBRID_GH2_GH4_PREDICTOR_H
12
13
14
     {\it \#include \ "predictor.h"}
     #include "gshare_predictor.h"
#include "nbit_predictor.h"
15
16
     {\it \#include \ "hybrid\_predictor.h"}
18
     class hybrid_GH2_GH4_predictor : public hybrid_predictor
19
20
     public:
21
22
23
             hybrid_GH2_GH4_predictor(int entries);
24
25
     };
26
27
28
    #endif
29
```

### gshare global history BHT=2

```
6
    _----*/
8
9
10
    #include "hybrid_GS_GH2_predictor.h"
11
12
13
    hybrid_GS_GH2_predictor::hybrid_GS_GH2_predictor(int entries): hybrid_predictor(entries)
14
15
            preds[0] = new gshare_predictor();
16
17
            preds[1] = new globalhistory_predictor(8192,2,2);
    }
18
1
    /* -.-.-.-.-.
     * File Name : hybrid_GS_GH2_predictor.h
2
     * Creation Date : 27-05-2012
3
    * Last Modified : Sun 27 May 2012 11:12:16 PM EEST
     * Created By : Greg Liras <gregliras@gmail.com>
5
6
     7
8
Q
10
     #ifndef HYBRID_GS_GH2_PREDICTOR H
11
12
    #define HYBRID_GS_GH2_PREDICTOR_H
13
    #include "predictor.h"
14
    {\it \#include \ "gshare\_predictor.h"}
15
    #include "nbit_predictor.h"
16
    #include "hybrid_predictor.h"
17
18
    class hybrid_GS_GH2_predictor : public hybrid_predictor
19
20
    public:
21
22
23
            hybrid_GS_GH2_predictor(int entries);
24
25
    };
26
27
28
    #endif
```

## gshare global history BHT=4

```
1
    * File Name : hybrid_GS_GH4_predictor.cpp
2
    * Creation Date : 27-05-2012
3
    * Last Modified : Sun 27 May 2012 11:14:51 PM EEST
5
    * Created By : Greg Liras <gregliras@gmail.com>
6
    _._._...*/
8
9
10
    #include "hybrid_GS_GH4_predictor.h"
11
12
13
    hybrid_GS_GH4_predictor::hybrid_GS_GH4_predictor(int entries): hybrid_predictor(entries)
14
15
           preds[0] = new gshare_predictor();
16
           preds[1] = new globalhistory_predictor(8192,2,4);
17
    }
18
1
2
    * File Name : hybrid\_GS\_GH4\_predictor.h
    * Creation Date : 27-05-2012
3
    * Last Modified : Sun 27 May 2012 11:15:00 PM EEST
4
    * Created By : Greg Liras <gregliras@gmail.com>
6
    _-----*/
8
```

```
10
       #ifndef HYBRID_GS_GH4_PREDICTOR_H
#define HYBRID_GS_GH4_PREDICTOR_H
11
12
13
      #include "predictor.h"
#include "gshare_predictor.h"
#include "nbit_predictor.h"
14
15
16
17
       #include "hybrid_predictor.h"
18
       class hybrid_GS_GH4_predictor : public hybrid_predictor
19
20
21
       public:
22
                  hybrid_GS_GH4_predictor(int entries);
23
24
25
26
       };
27
28
      #endif
29
```