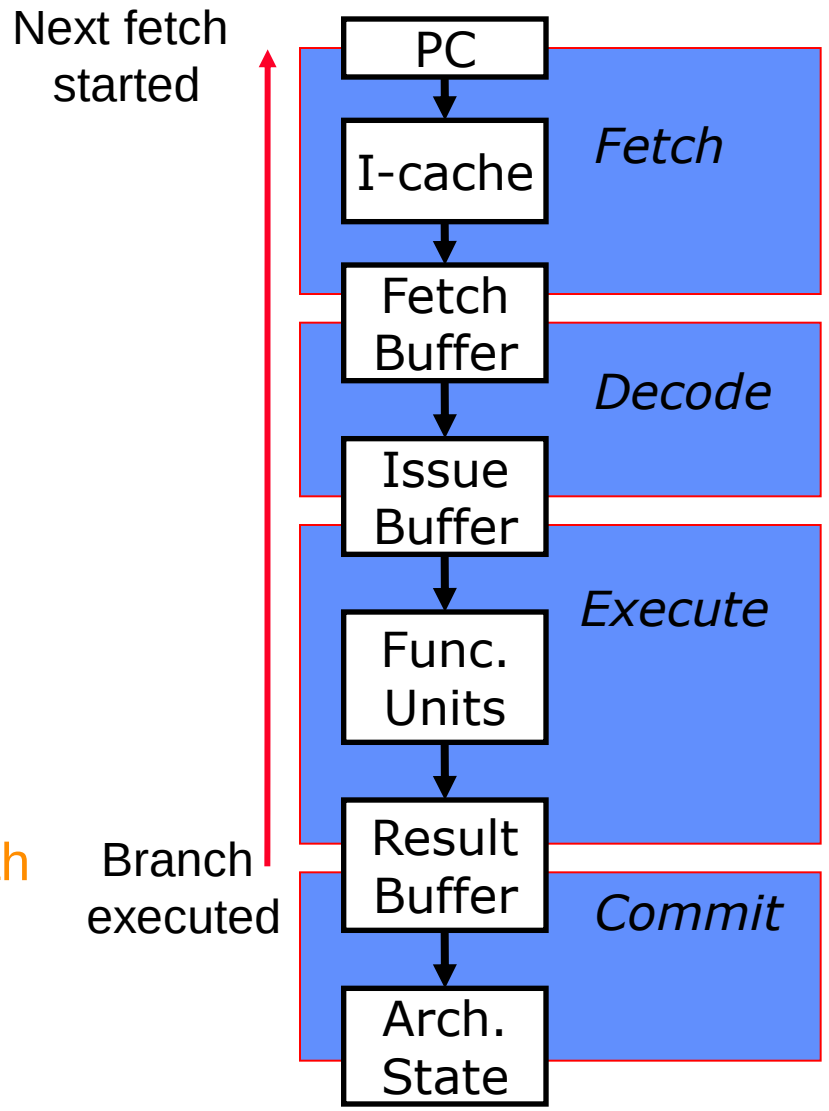
Μέθοδοι Πρόβλεψης Διακλαδώσεων (Branch Prediction Mechanisms)

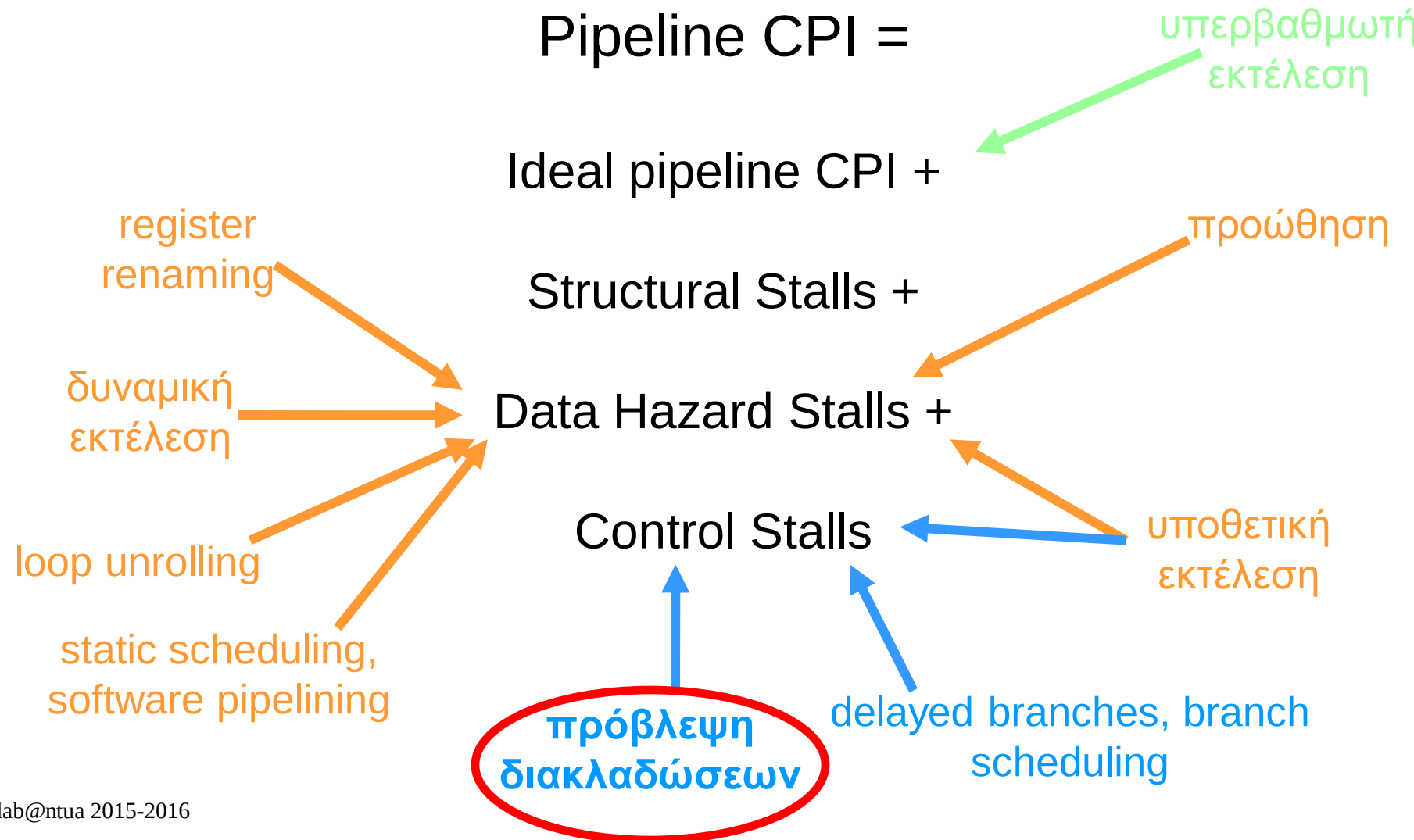
Εντολές Διακλάδωσης

- Περίπου 20% των εντολών είναι εντολές διακλάδωσης
- Πολλά στάδια μεταξύ υπολογισμού του επόμενου PC και εκτέλεσης του branch (για σύγχρονους επεξεργαστές μπορεί και >10!)
- Εισαγωγή stalls και επομένως μείωση του ρυθμού ανάγνωσης και εκτέλεσης εντολών

#bubbles ~= pipeline depth X loop length



Τεχνικές βελτίωσης του CPI



Τεχνικές Αντιμετώπισης
Control Dependencies

- Stall the pipeline
- Κάνε κάτι χρήσιμο (branch delay slots)
- Διαγραφή control-flow instructions (predicated execution)
- Κάνε κάτι άλλο (fine-grained multithreading)
- Κάνε τα όλα (multipath execution)
- Πρόβλεψη

Εντολές Άλματος

- Χρειαζόμαστε 2 πληροφορίες
  - Αν θα εκτελεστεί το άλμα ή όχι (taken or not taken)
  - Αν εκτελεστεί ποιος είναι ο προορισμός (target PC)

Είδος Άλματος	Απόφαση	Προορισμός
Direct Jumps Function Calls	Always Taken	Υπολογίζεται εύκολα
Conditional Branches	???	Υπολογίζεται εύκολα
Indirect Jumps Function returns	Always Taken	Υπολογίζεται δύσκολα

Πρόβλεψη Απόφασης

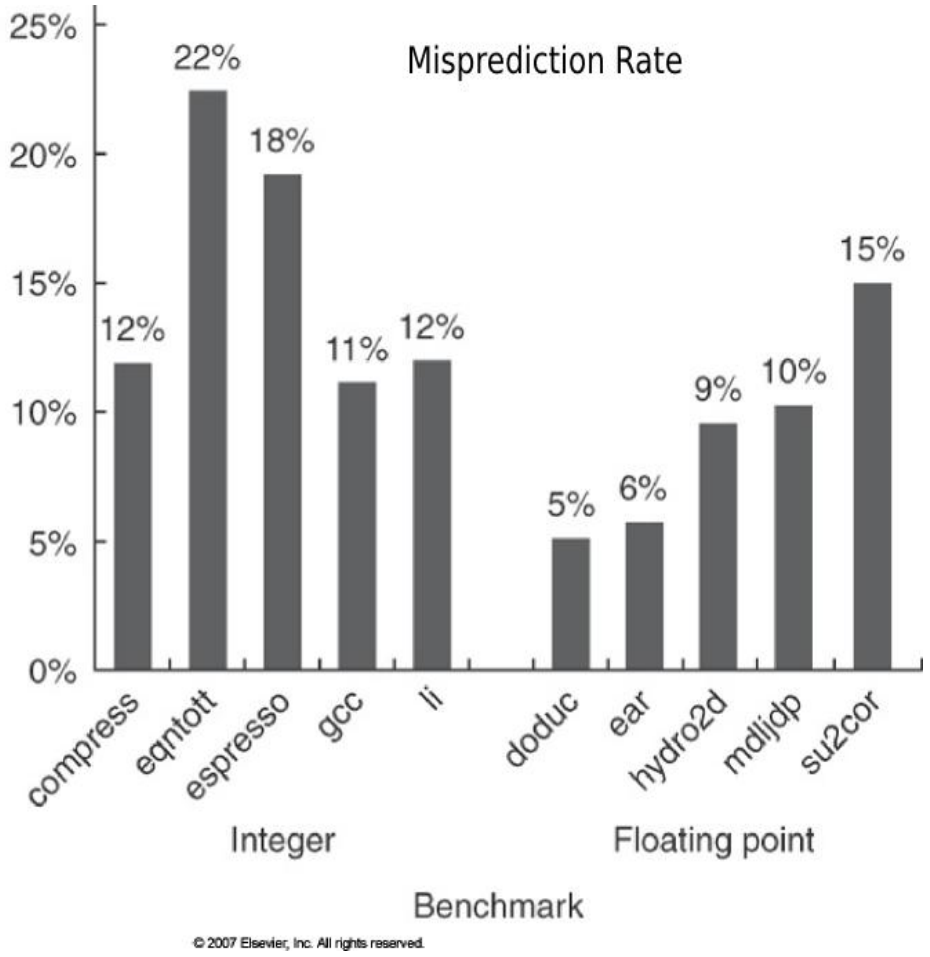
- Απαιτείται για εντολές διακλάδωσης υπό συνθήκη
  - Η πλειοψηφία των εντολών διακλάδωσης είναι υπό συνθήκη
- 2 είδη τεχνικών πρόβλεψης
  - Στατικές
  - Δυναμικές
- Απαιτείται extra hardware
  - Αποθήκευση χρήσιμων πληροφοριών για βελτίωση της ακρίβειας των προβλέψεων (branch history tables, branch target buffers, etc)
  - Μηχανισμός ανάνηψης σε περίπτωση λανθασμένης πρόβλεψης

Στατικές Τεχνικές Πρόβλεψης

- Branch not taken (NT)
  - Εύκολη υλοποίηση
  - Σε ένα loop σωστή πρόβλεψη μόνο στην τελευταία εκτέλεση
  - Misprediction rate ~60%-70%
- Branch taken (T)
  - Πιο πολύπλοκο hardware
  - Σε ένα loop λάθος πρόβλεψη μόνο στην τελευταία εκτέλεση
  - Average misprediction rate 34% (SPEC benchmarks)
- BTFNT
  - Άλματα προς τα πίσω (αρνητικό offset ως προς το PC) προβλέπεται ότι θα εκτελεστούν (Backwards taken)
  - Άλματα προς τα εμπρός (θετικό offset ως προς το PC) προβλέπεται ότι δε θα εκτελεστούν (Forwards not taken)
  - π.χ. χρησιμοποιείται στον Intel Pentium 4 σε περίπτωση που αποτύχει ο μηχανισμός δυναμικής πρόβλεψης

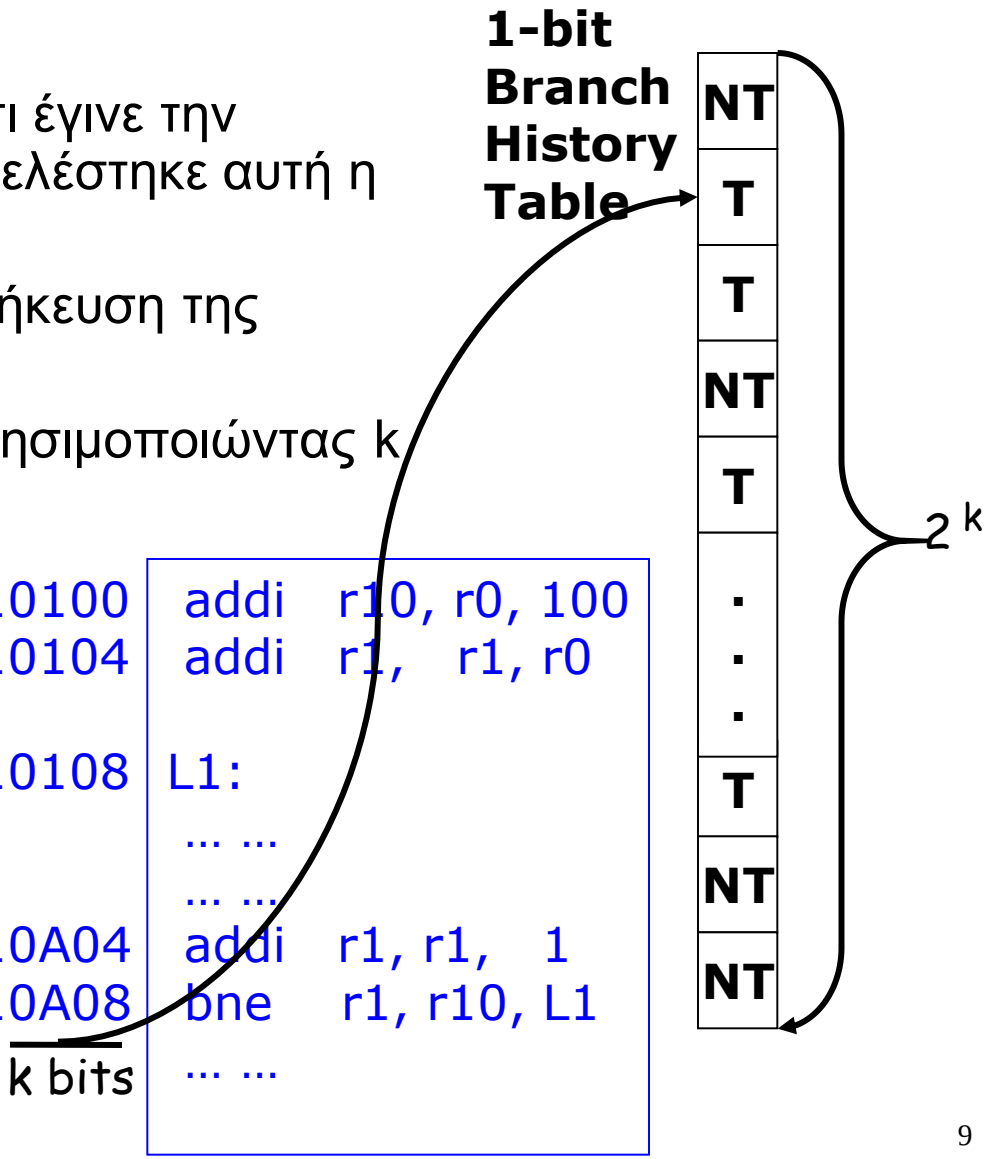
Στατικές Τεχνικές Πρόβλεψης

- Profiling
  - Εκτέλεση προγράμματος και καταγραφή στατιστικών
  - Ο compiler τα χρησιμοποιεί για να βοηθήσει το hardware να κάνει σωστή πρόβλεψη (π.χ. αν μια εντολή διακλάδωσης εκτελείται πάνω από τις μισές φορές κατά τη διάρκεια του profiling τότε η πρόβλεψη είναι Τ)
  - Εύκολη υλοποίηση
  - Τα δεδομένα του profiling και της κανονικής εκτέλεσης μπορεί να είναι πολύ διαφορετικά. Επομένως λάθος προβλέψεις
- Program-based
- Programmer-based (C likely/unlikely)



Δυναμικές Τεχνικές Πρόβλεψης

- 1-bit predictor
  - Η πρόβλεψη βασίζεται στο τι έγινε την προηγούμενη φορά που εκτελέστηκε αυτή η εντολή διακλάδωσης
  - Χρήση πίνακα για την αποθήκευση της απόφασης
  - Προσπέλαση του πίνακα χρησιμοποιώντας k bits του PC
    - » Aliasing



Παράδειγμα 1-bit predictor

```
0x108:  for(i=0; i < 100000; i++)  {
                ......................
0x144:      if( ( i % 100) == 0 )
                callA( );
0x150:      if( (i & 1) == 1)
                callB( );
}
```

Πρόβλεψη (108):

0

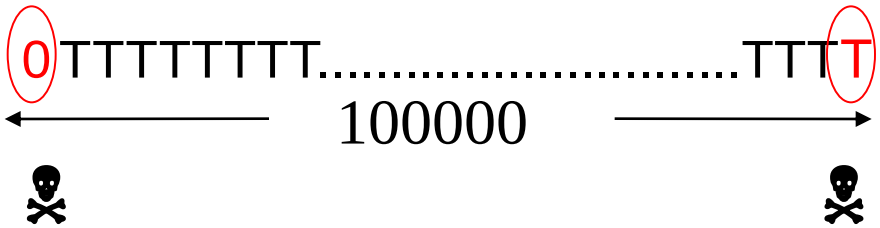
Απόφαση (108):

T

Παράδειγμα 1-bit predictor

```
0x108:  for(i=0; i < 100000; i++)  {
                    ..........................
0x144:      if( ( i % 100) == 0 )
                    callA( );
0x150:      if( (i & 1) == 1)
                    callB( );
}
```

Πρόβλεψη (108):



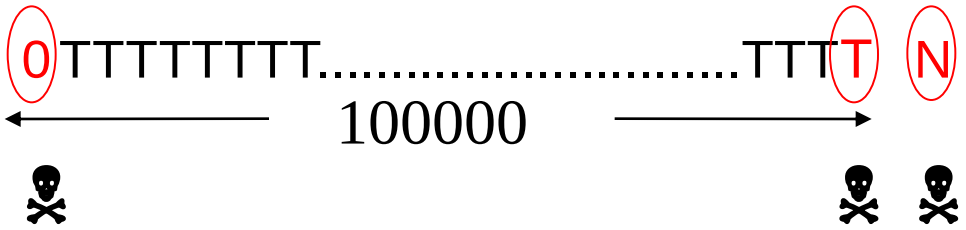
Απόφαση (108):

TTTTTTTTT.....TTTN

Παράδειγμα 1-bit predictor

```
0x108:  for(i=0; i < 100000; i++)  {
                    ..........................
0x144:      if( ( i % 100) == 0 )
                    callA( );
0x150:      if( (i & 1) == 1)
                    callB( );
}
```

Πρόβλεψη (108):



Απόφαση (108):

TTTTTTTTT.....TTTN T

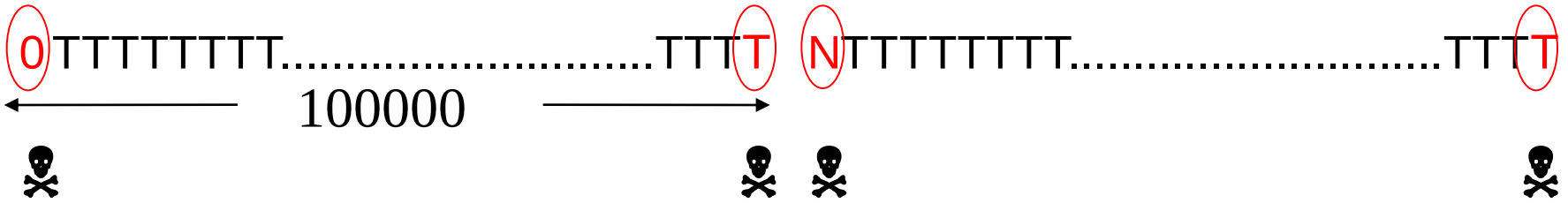
Παράδειγμα 1-bit predictor

```
0x108:  for(i=0; i < 100000; i++)  {
                  ..........................
0x144:      if( ( i % 100) == 0 )
                  callA( );
0x150:      if( (i & 1) == 1)
                  callB( );
}
```

Misprediction = 2/100000

Prediction Rate = 99.998%

Πρόβλεψη (108):



Απόφαση (108):

TTTTTTTTT.....TTTN TTTTTTTTT.....TTTN

Παράδειγμα 1-bit predictor

```
0x108:  for(i=0; i < 100000; i++)  {
                  ..........................
0x144:      if( ( i % 100) == 0 )
                  callA( );
0x150:      if( (i & 1) == 1)
                  callB( );
}
```

```
DIV     R2,#100
MFHI    R1
BNEZ    R1,0x150
JMP     FUNA
```

Πρόβλεψη (144):

0

Απόφαση (144):

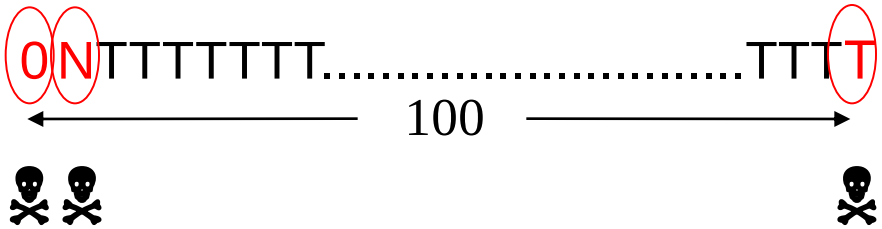
N

Παράδειγμα 1-bit predictor

```
0x108:  for(i=0; i < 100000; i++)  {
                  ..........................
0x144:      if( ( i % 100) == 0 )
                  callA( );
0x150:      if( (i & 1) == 1)
                  callB( );
}
```

```
DIV     R2,#100
MFHI    R1
BNEZ    R1,0x150
JMP     FUNA
```

Πρόβλεψη (144):



Απόφαση (144):

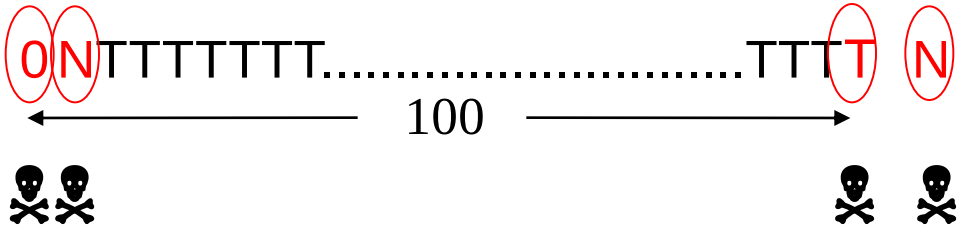
NTTTTTTTT.....TTTN

Παράδειγμα 1-bit predictor

```
0x108:  for(i=0; i < 100000; i++)  {
                  ..........................
0x144:      if( ( i % 100) == 0 )
                  callA( );
0x150:      if( (i & 1) == 1)
                  callB( );
}
```

```
DIV     R2,#100
MFHI    R1
BNEZ    R1,0x150
JMP     FUNA
```

Πρόβλεψη (144):



Απόφαση (144):

NTTTTTTTT.....TTTN T

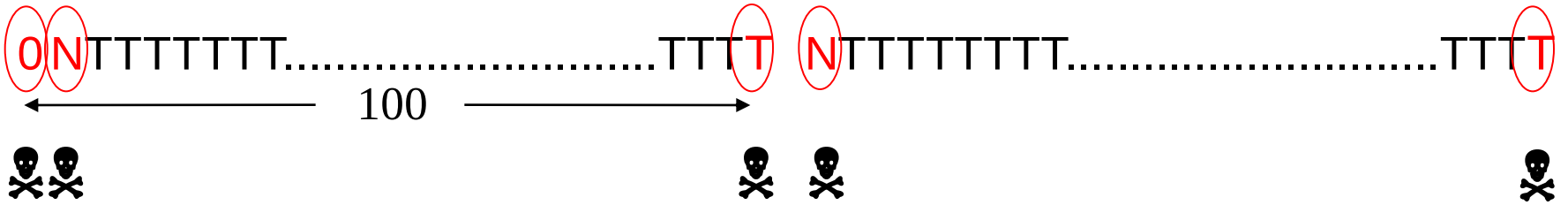
Παράδειγμα 1-bit predictor

```
0x108:  for(i=0; i < 100000; i++)  {
                  ..........................
0x144:      if( ( i % 100) == 0 )
                  callA( );
0x150:      if( (i & 1) == 1)
                  callB( );
}
```

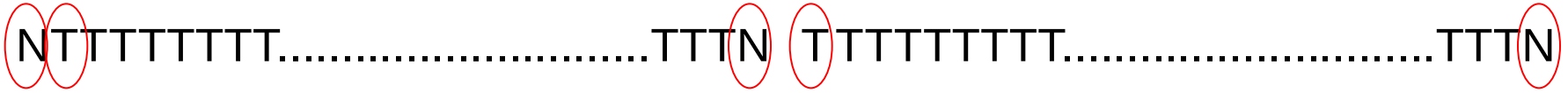
Misprediction = 2/100

Prediction Rate = 98%

Πρόβλεψη (144):



Απόφαση (144):



Παράδειγμα 1-bit predictor

```
0x108:  for(i=0; i < 100000; i++)  {
                  ..........................
0x144:      if( ( i % 100) == 0 )
                  callA( );
0x150:      if( (i & 1) == 1)
                  callB( );
}
```

```
AND     R1,R2,#1
SUB     R1,#1
BNEZ    R1,ENDLOOP
JMP     FUNB
```

Πρόβλεψη (150):

0

Απόφαση (150):

T

Παράδειγμα 1-bit predictor

```
0x108:  for(i=0; i < 100000; i++)  {
                  ..........................
0x144:      if( ( i % 100) == 0 )
                  callA( );
0x150:      if( (i & 1) == 1)
                  callB( );
}
```

```
AND     R1,R2,#1
SUB     R1,#1
BNEZ    R1,ENDLOOP
JMP     FUNB
```

Πρόβλεψη (150):

0 T

☠ ☠

Απόφαση (150):

T N

Παράδειγμα 1-bit predictor

```
0x108:  for(i=0; i < 100000; i++)  {
                  ......................
0x144:      if( ( i % 100) == 0 )
                  callA( );
0x150:      if( (i & 1) == 1)
                  callB( );
}
```

```
AND     R1,R2,#1
SUB     R1,#1
BNEZ    R1,ENDLOOP
JMP     FUNB
```

Πρόβλεψη (150):

0 T N

Απόφαση (150):

T N T

Παράδειγμα 1-bit predictor

```
0x108:  for(i=0; i < 100000; i++)  {
                  ..........................
0x144:      if( ( i % 100) == 0 )
                      callA( );
0x150:      if( (i & 1) == 1)
                      callB( );
}
```

Misprediction = 1/1

Prediction Rate = 0%

Πρόβλεψη (150):

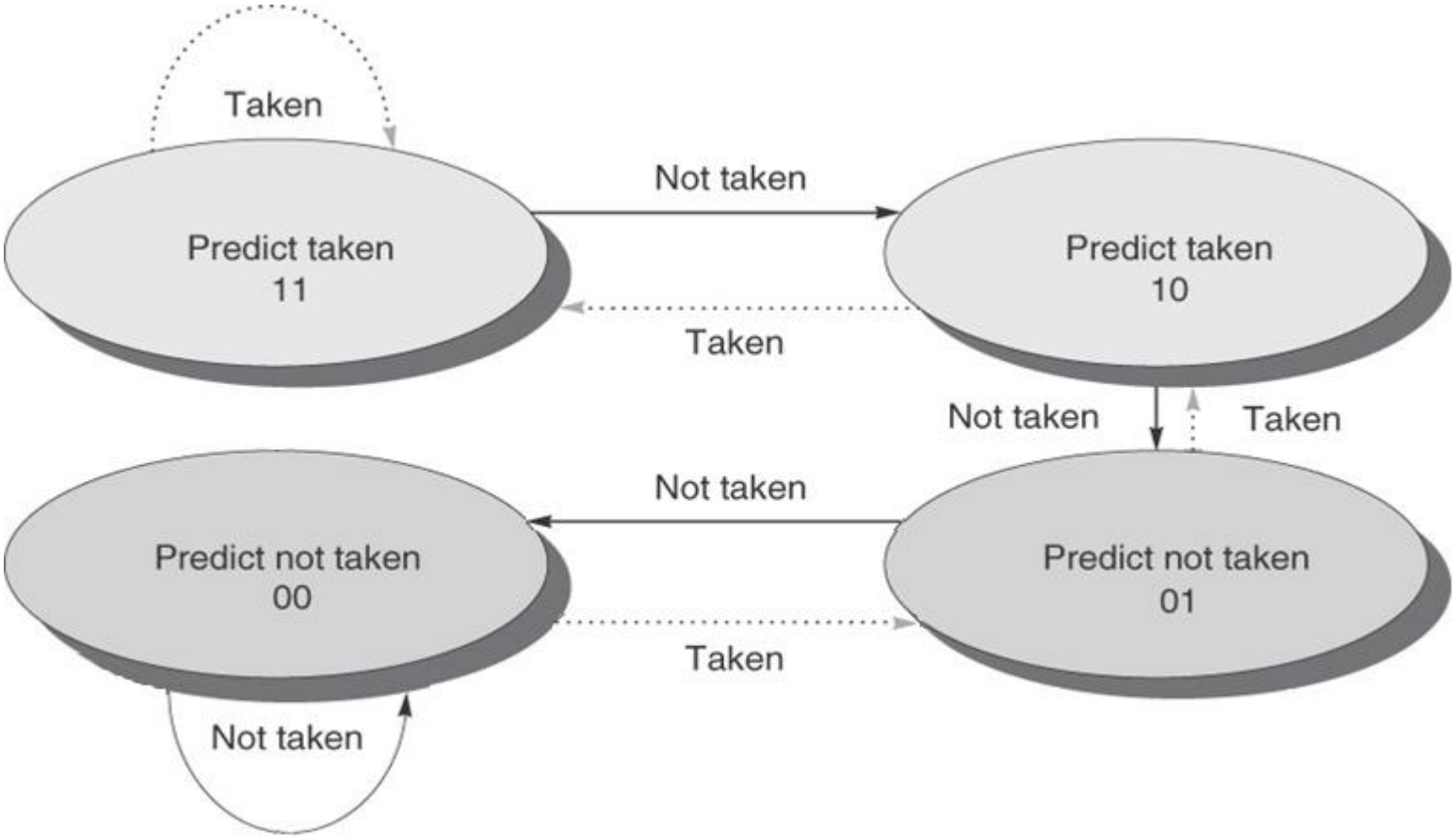
0 T N T N T N T N T N T N T N T N T N T N T N T N T N

Απόφαση (150):

T N T N T N T N T N T N T N T N T N T N T N T N T N

Δυναμικές Τεχνικές Πρόβλεψης

- 2-bit predictor



Παράδειγμα 2-bit predictor

```
0x108:  for(i=0; i < 100000; i++)  {
                ....................
0x144:      if( ( i % 100) == 0 )
                callA( );
0x150:      if( (i & 1) == 1)
                callB( );
}
```

0,1 :Predict Not Taken

2,3 :Predict Taken

Πρόβλεψη (108):

1

Απόφαση (108):

T

Παράδειγμα 2-bit predictor

```
0x108:  for(i=0; i < 100000; i++)  {
                    ..........................
0x144:      if( ( i % 100) == 0 )
                    callA( );
0x150:      if( (i & 1) == 1)
                    callB( );
}
```

0,1 :Predict Not Taken

2,3 :Predict Taken

Πρόβλεψη (108):

1 23333333.....3333

100000


Απόφαση (108):

TTTTTTTTT.....TTTN

Παράδειγμα 2-bit predictor

```
0x108:  for(i=0; i < 100000; i++)  {
                    ..........................
0x144:      if( ( i % 100) == 0 )
                    callA( );
0x150:      if( (i & 1) == 1)
                    callB( );
}
```

0,1 : Predict Not Taken

2,3 : Predict Taken

Πρόβλεψη (108):

1 23333333.....3333 2

100000

☠ ☠ √

Απόφαση (108):

TTTTTTTTT.....TTTN T

Παράδειγμα 2-bit predictor

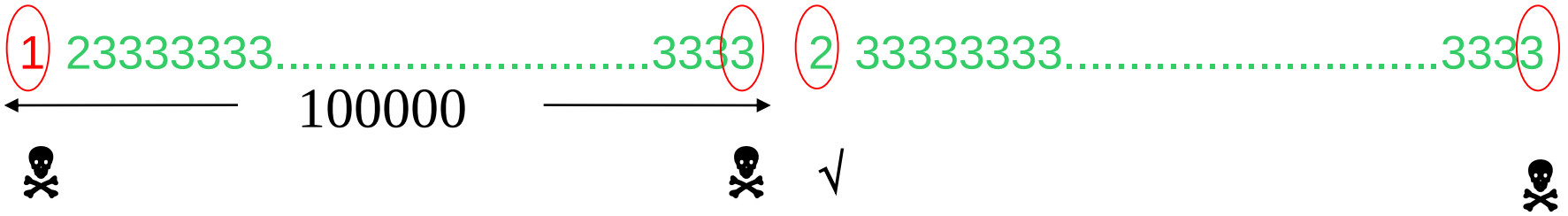
```
0x108:  for(i=0; i < 100000; i++)  {
                  ..........................
0x144:      if( ( i % 100) == 0 )
                  callA( );
0x150:      if( (i & 1) == 1)
                  callB( );
}
```

Misprediction ~= 1 per N branches

0x108 Prediction Rate = 99.999%
0x144 Prediction Rate = 99%
0x150 Prediction Rate = 50%

0,1 :Predict Not Taken
2,3 :Predict Taken

Πρόβλεψη (108):

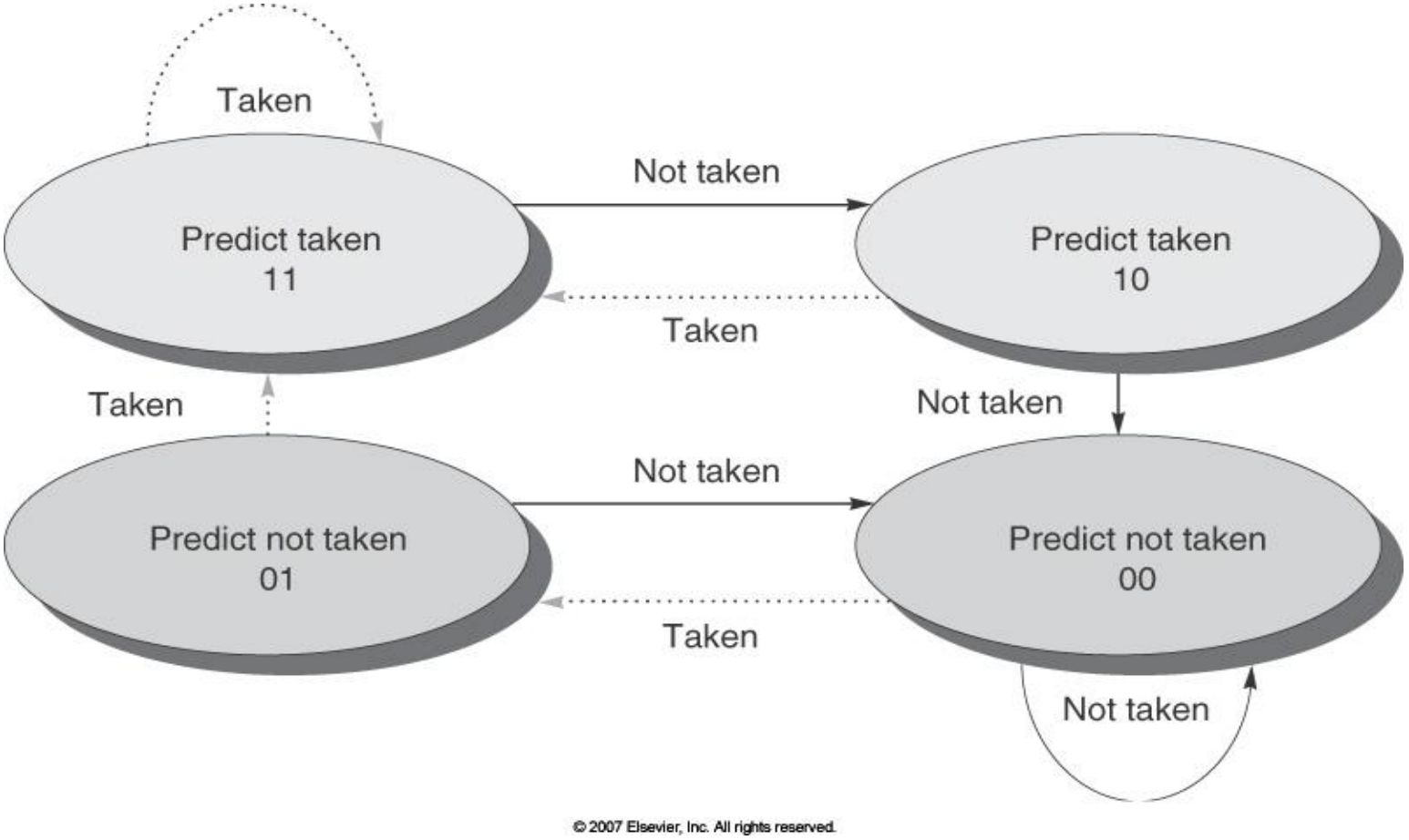


Απόφαση (108):

TTTTTTTTT.....TTTN TTTTTTTTT.....TTTN

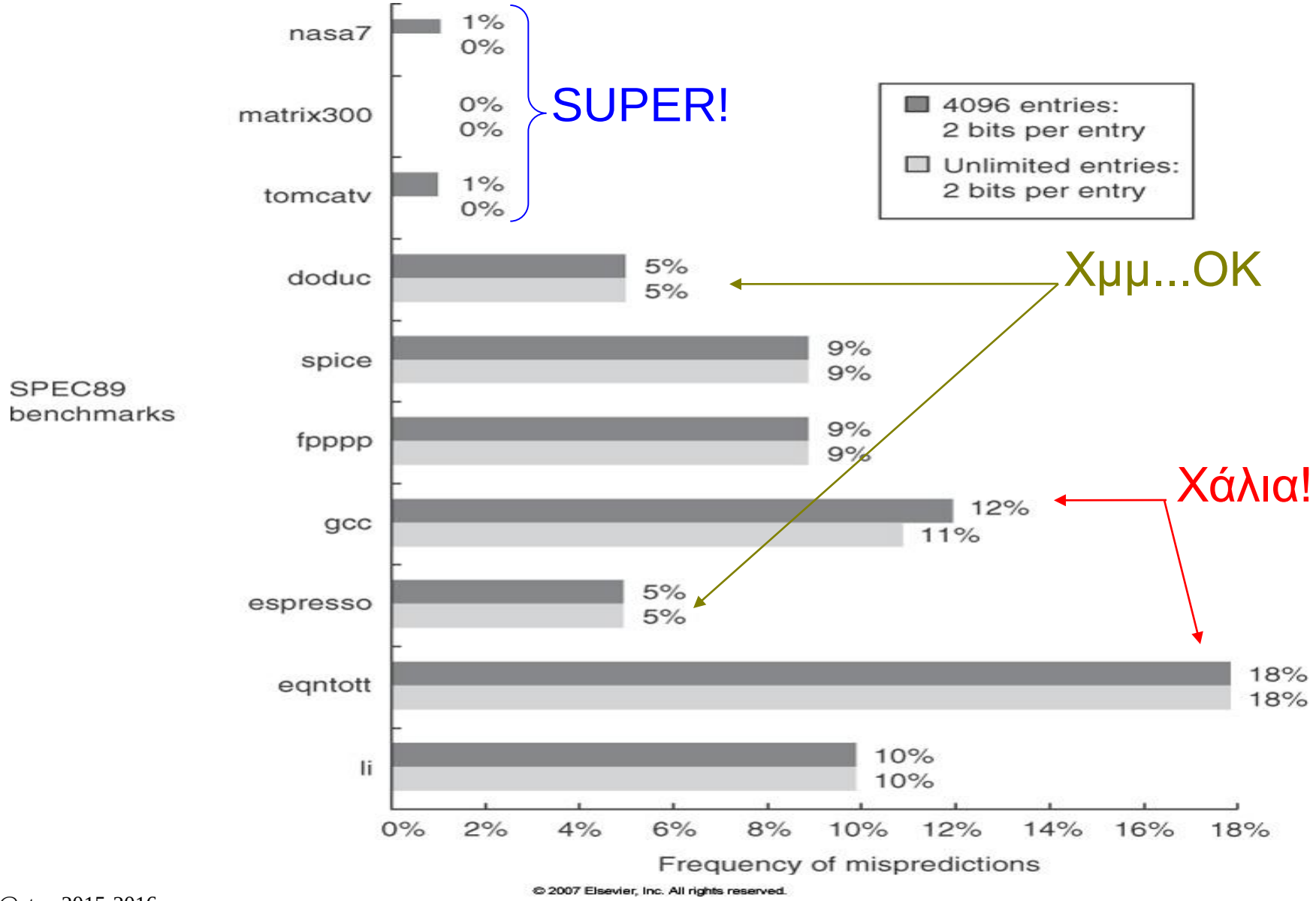
Δυναμικές Τεχνικές Πρόβλεψης

- Άλλος 2-bit predictor



- 2^{20} δυνατά FSMs → 5248 “ενδιαφέροντα” [Nair, 1992]

Ακρίβεια Πρόβλεψης για 2-bits predictor

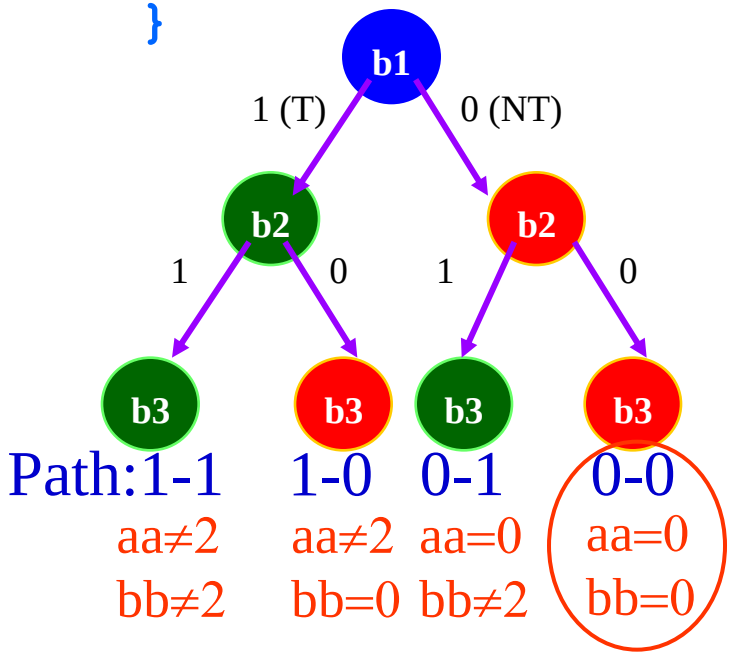


Δυναμικές Τεχνικές Πρόβλεψης

- Χρονική Συσχέτιση (Temporal Correlation)
  - Όλες οι προηγούμενες τεχνικές προβλέπουν το αποτέλεσμα μιας εντολής διακλάδωσης με βάση τις αποφάσεις που πάρθηκαν για την ίδια διακλάδωση σε προηγούμενες εκτελέσεις.
- Τοπική Συσχέτιση (Spatial Correlation)
  - Πρόβλεψη μιας εντολής διακλάδωσης με βάση τη συμπεριφορά άλλων εντολών διακλάδωσης που προηγούνται στη δυναμική ροή του προγράμματος

Παράδειγμα

```
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)
```

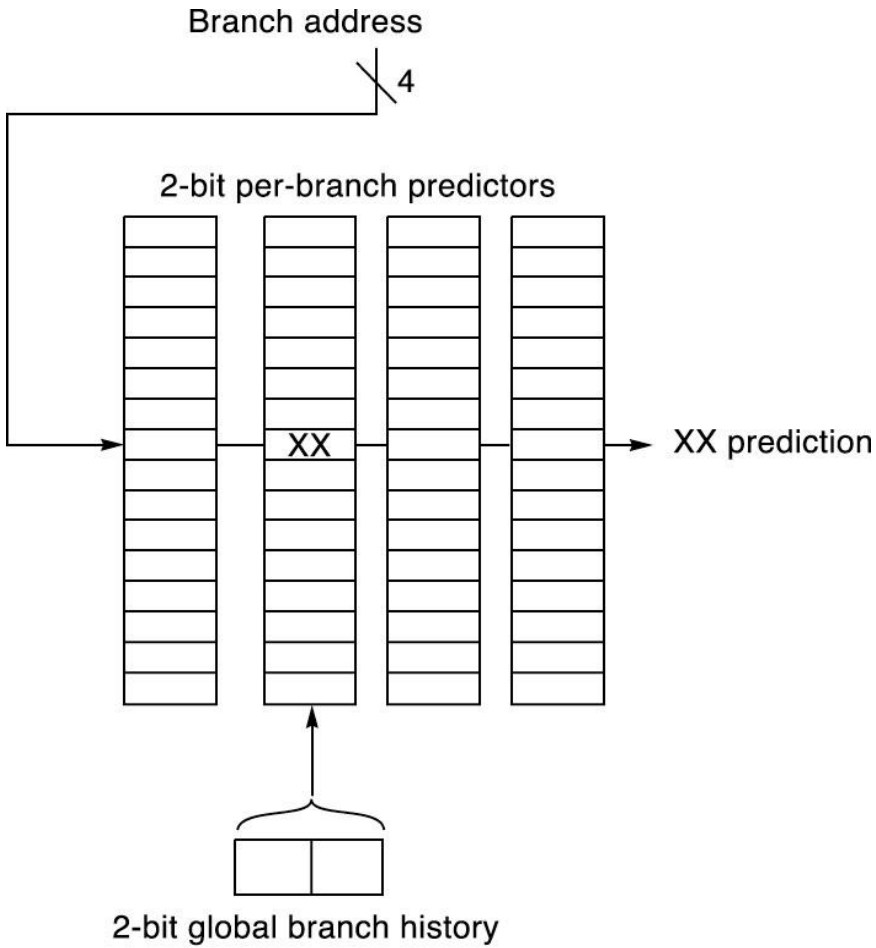
Αν b1 και b2 NT (Not Taken) τότε b3 T (Taken) !

Correlating/Two-level Predictors

- Γενική περίπτωση : (m,n) predictor
  - m τελευταίες εντολές διακλάδωσης
  - επιλογή ενός από 2^m predictors
  - Κάθε predictor είναι n-bits
- Ο 2-bit predictor είναι ένας (0,2) predictor αφού δεν χρησιμοποιεί την ιστορία των άλλων εντολών διακλάδωσης
- Απλή υλοποίηση
  - Branch History Register (BHR) : m-bit shift register για να καταγράφει τη συμπεριφορά των τελευταίων m εντολών διακλάδωσης
  - Pattern History Table (PHT) : Ο πίνακας που αποθηκεύονται οι predictors

Global-History Two-Level Predictor

- (2,2) predictor
- 64 entries
- 4 low order bits PC
- 2 bits global history

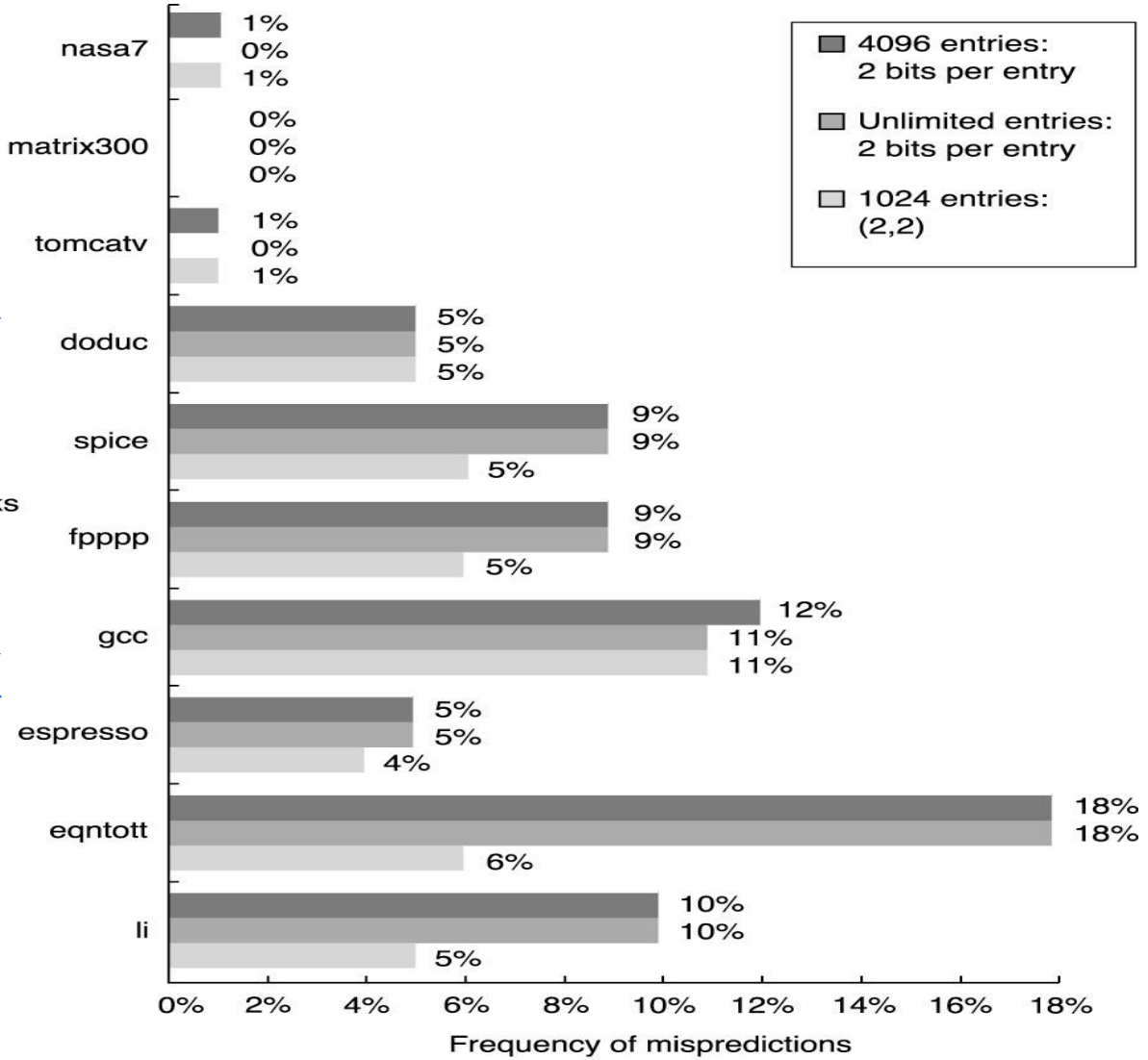


Σύγκριση

- (0,2) predictor με 4096 εγγραφές (8K bits)

vs

- (2,2) predictor με 1024 εγγραφές (8K bits)

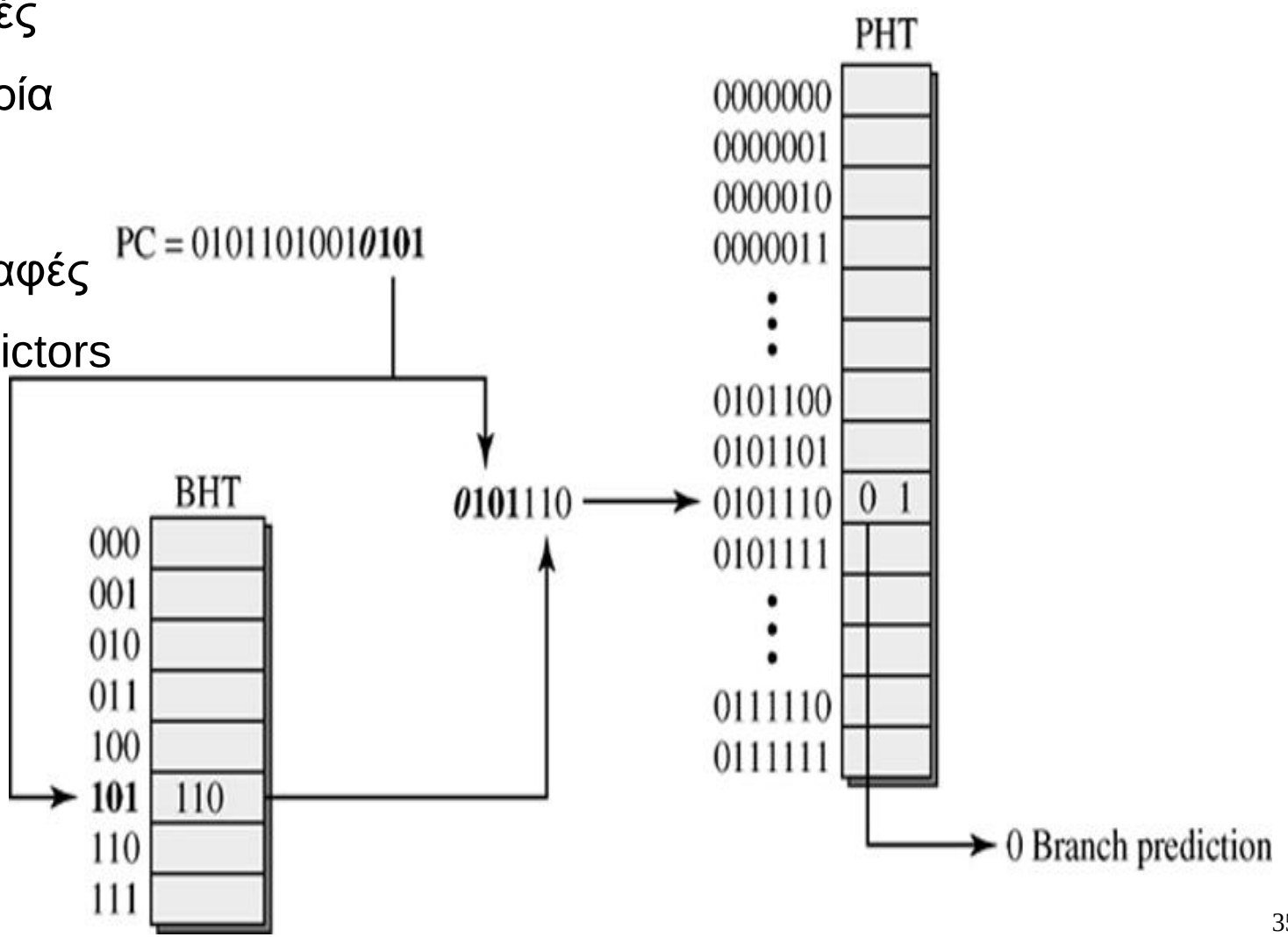


Local-History Two-Level Predictor

- Αντί για τις m τελευταίες εντολές διακλάδωσης, παρακολουθούμε τις m τελευταίες εκτελέσεις της συγκεκριμένης εντολής
- Ο BHR αντικαθίσταται από τον BHT (Branch History Table)
  - 1 BHR ανά εντολή διακλάδωσης
- Ο global-history predictor αποτελεί ουσιαστικά υποπερίπτωση, όπου ο BHT έχει μόνο μια εγγραφή

Local-History Two-Level Predictor

- BHT
  - 8 εγγραφές
  - 3-bit ιστορία
- PHT
  - 128 εγγραφές
  - 2-bit predictors

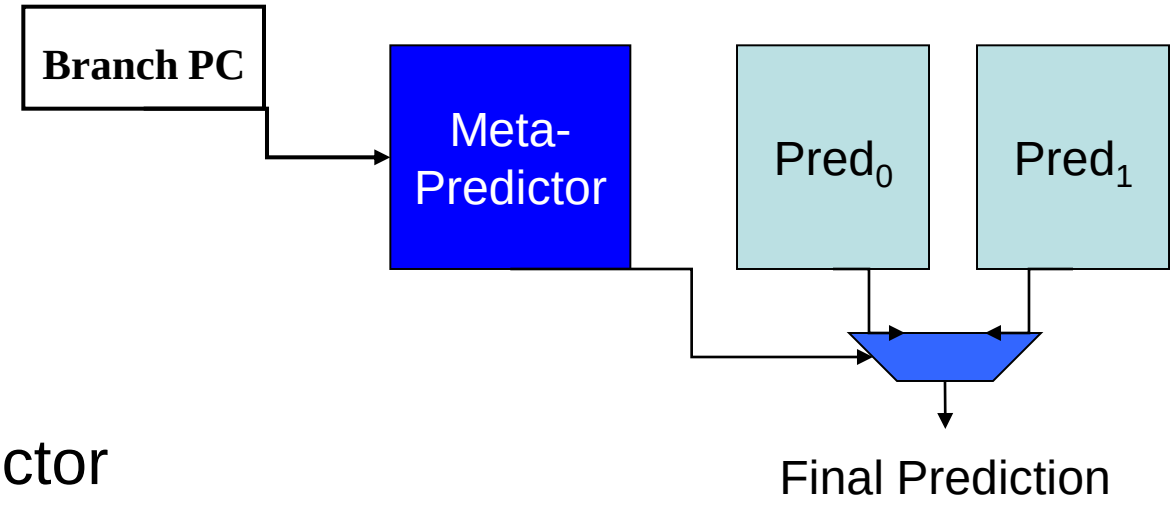


Tournament Predictors

- Δεν υπάρχει τέλειος predictor
  - Διαφορετικές εντολές άλματος παρουσιάζουν διαφορετική συμπεριφορά
- ΙΔΕΑ ??????
- Να κατασκευάσουμε ένα predictor που θα μαντεύει ποιος predictor μπορεί να μαντέψει ακριβέστερα το αποτέλεσμα ενός άλματος!

Tournament Hybrid Predictor

- meta-predictor
  - 2-bit μετρητές
  - 0,1 χρησιμοποιείται ο P_0
  - 2,3 χρησιμοποιείται ο P_1
- Η τιμή του meta-predictor ενημερώνεται μόνο όταν οι δυο predictors κάνουν διαφορετική πρόβλεψη
- $Pred_0$, $Pred_1$
  - Συνδυασμοί των προηγουμένων συστημάτων

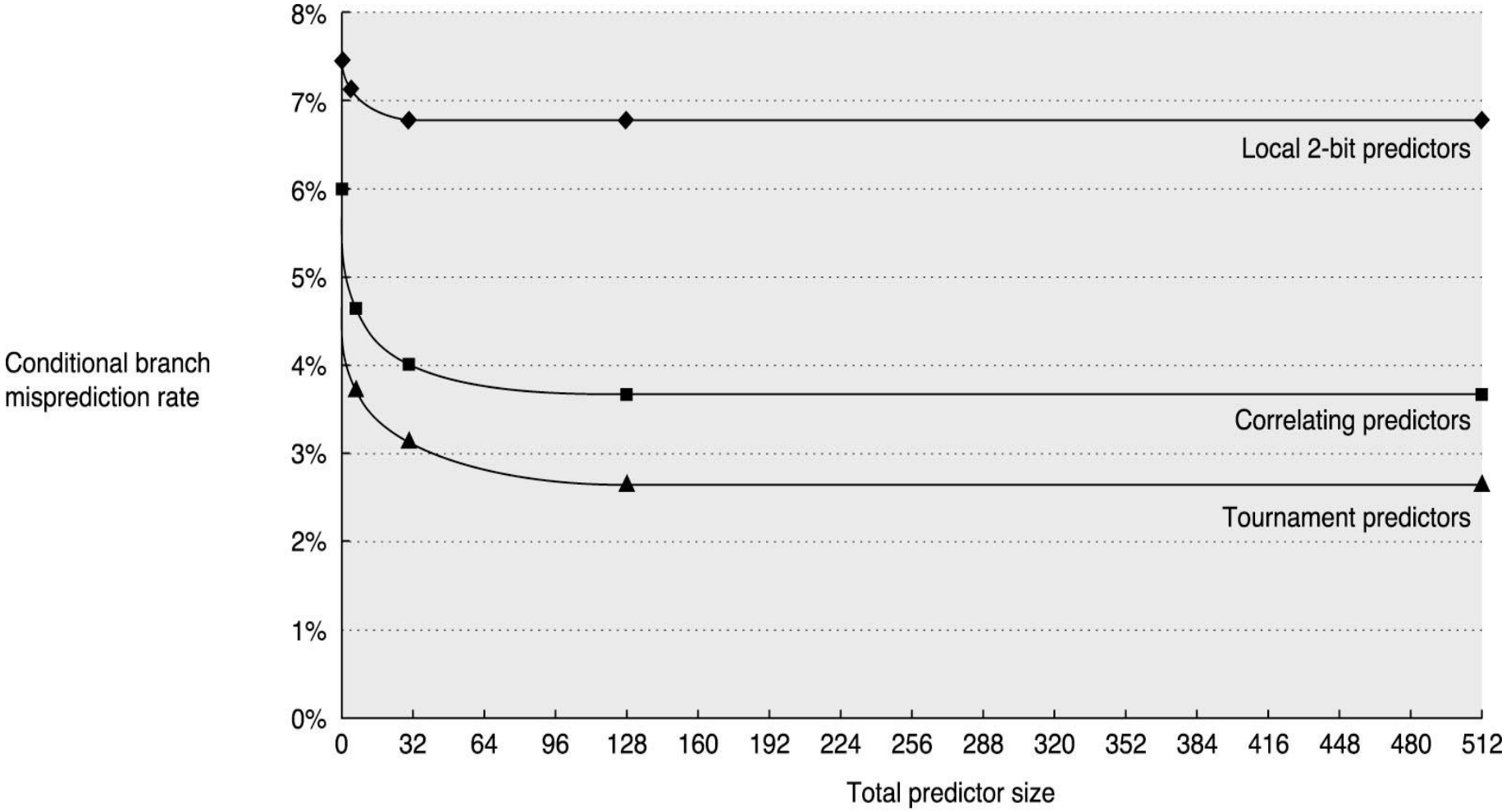


$Pred_0$	$Pred_1$	Meta Update
Λάθος	Λάθος	---
Λάθος	Σωστή	+1
Σωστή	Λάθος	-1
Σωστή	Σωστή	---

Παράδειγμα: Alpha 21264

- Meta-predictor
  - 4K εγγραφές
  - κάθε εγγραφή είναι ένας 2-bit predictor
  - προσπέλαση με βάση το PC της εντολής διακλάδωσης
- $Pred_0$: Local-history two-level predictor
  - BHT: 1K 10-bit εγγραφές
  - PHT: 1K 3-bit predictors
- $Pred_1$: Global-history two-level predictor
  - PHT: 4K 2-bit predictors
- Σύνολο : 29K bits
- SPECfp95 : misprediction = 1 / 1000 instructions
- SPECint95: misprediction = 11.5/1000 instructions

Σύγκριση Δυναμικών Τεχνικών Πρόβλεψης



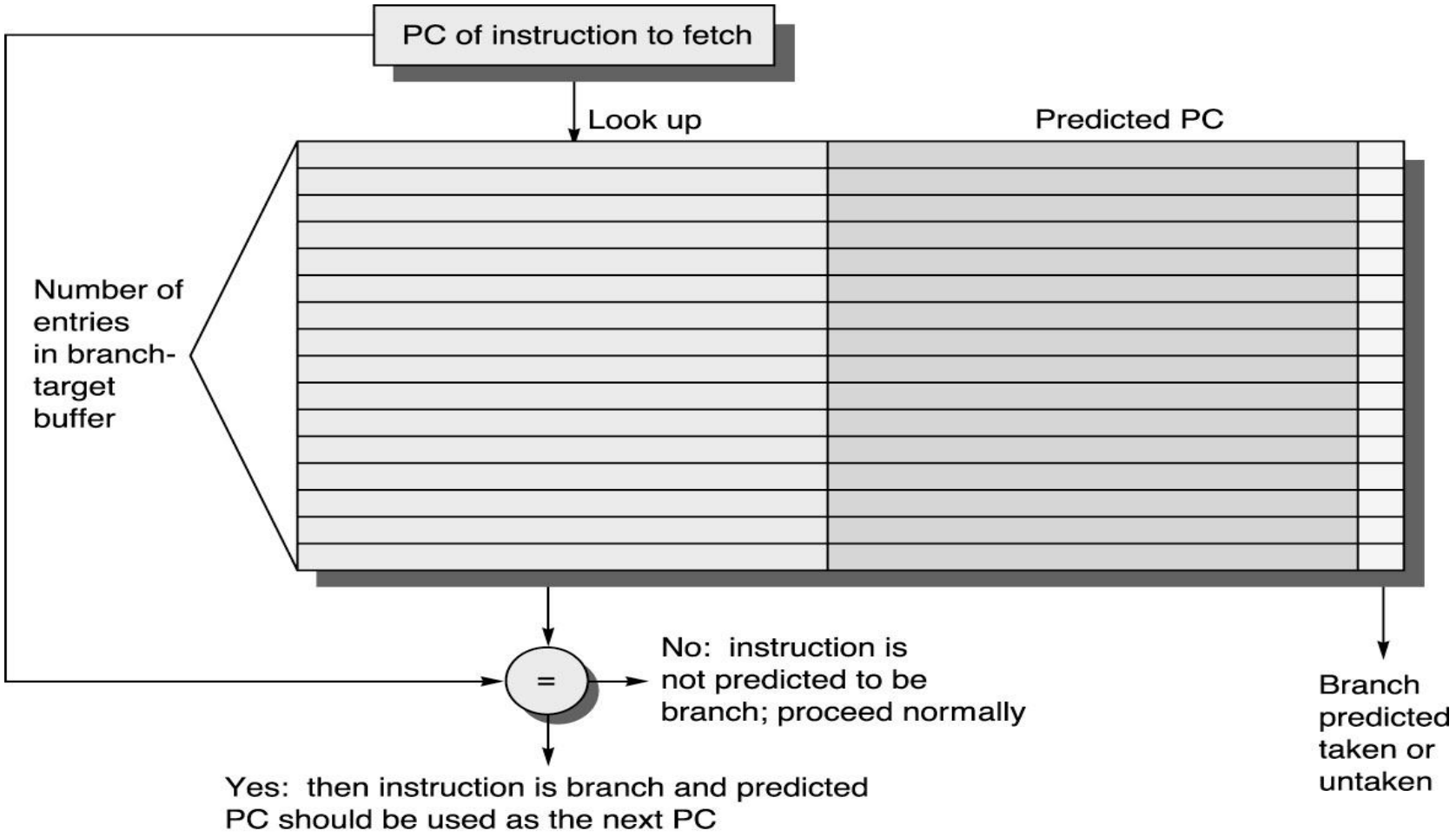
Πρόβλεψη Προορισμού

- Όλα τα προηγούμενα συστήματα προβλέπουν μόνο το ποιο μονοπάτι μιας εντολής διακλάδωσης θα ακολουθηθεί
- Χρειάζεται όμως και ο προορισμός-στόχος (target)
  - Not taken: Προορισμός = PC + instruct_word_size
  - Taken : Προορισμός = ???
    - » Άμεσος : PC + offset
    - » Έμμεσος : register_value + offset (π.χ. Object-oriented programs, subroutines returns, dynamically linked libraries)
- Για να διατηρήσουμε υψηλό throughput πρέπει στο τέλος κάθε κύκλου να γνωρίζουμε το επόμενο PC
- Για κάποια άλματα με έμμεσο προορισμό, γίνεται γνωστός μετά το EX
- Ακόμα και για τα υπόλοιπα πρέπει να περιμένουμε μέχρι το τέλος του ID

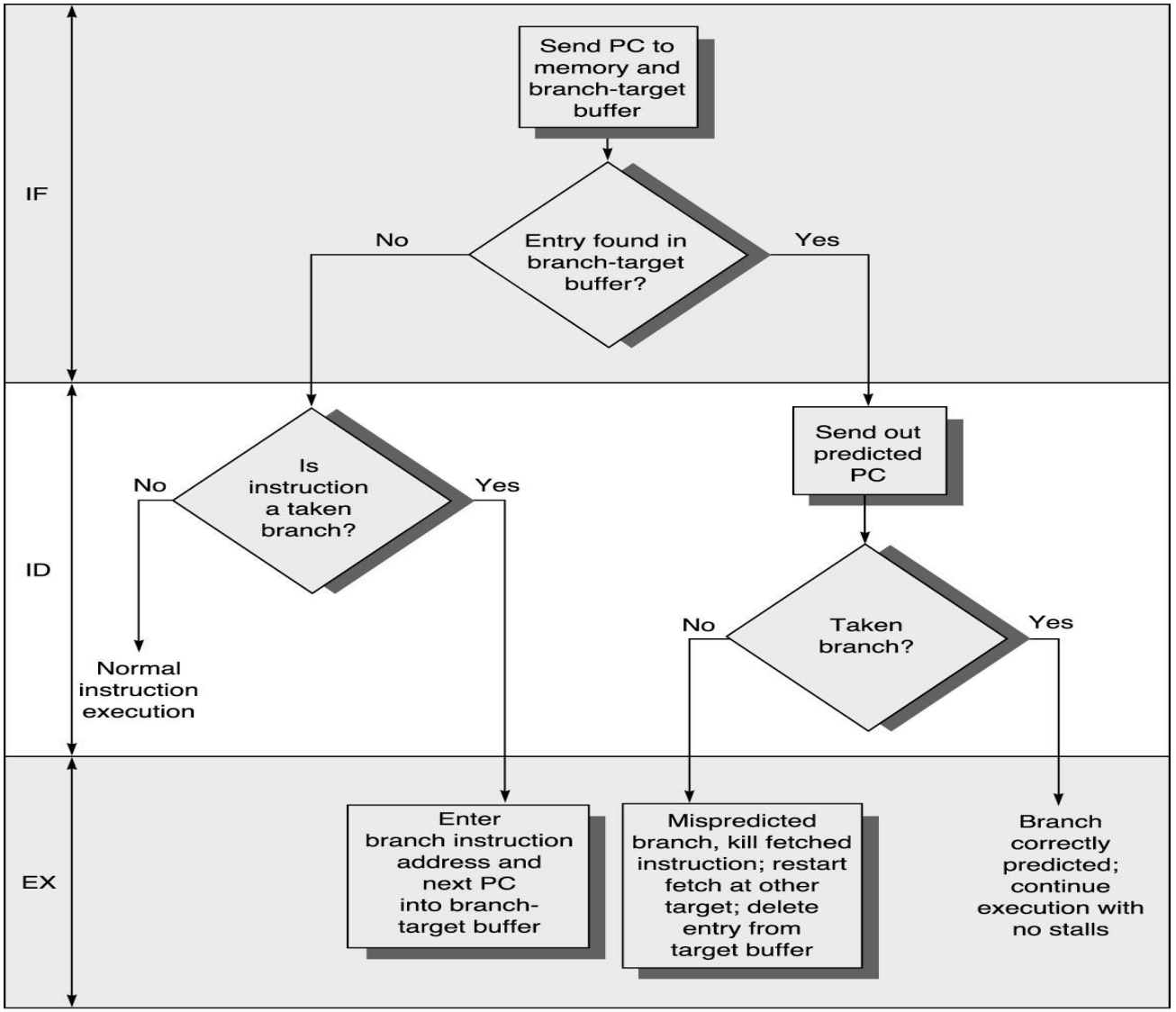
Branch-Target Buffer (BTB)

- Μια μικρή cache (direct-mapped / associative)
- Αποθηκεύει τον προορισμό (target) της εντολής άλματος
- Προσπέλαση κατά τη διάρκεια του IF, ώστε την ώρα που φέρνουμε μια εντολή ταυτόχρονα προβλέπουμε από που θα χρειαστεί να φέρουμε την επόμενη
- Περιέχει
  - Instruction Address
  - Predicted PC
- Αποθηκεύουμε μόνο άλματα που έχουν εκτελεστεί (taken branches και jumps)

Branch-Target Buffer



Χρήση BTB



Return Address Stack (RAS)

- SPEC89 : 85% των έμμεσων αλμάτων είναι function returns

- Προορισμός
  - Δύσκολος να υπολογιστεί. Γίνεται γνωστός μετά το EX.
  - Δύσκολα μπορεί να προβλεφθεί με τον BTB, μιας και ένα function μπορεί να κληθεί από πολλά διαφορετικά σημεία.

Return Address Stack (RAS)

- Ο προορισμός ενός return είναι **ΠΑΝΤΑ** η επόμενη διεύθυνση της τελευταίας εντολής call

- Χρήση ενός stack (FILO)
  - Εκτέλεση call → push address into RAS
  - Εκτέλεση return → pop address into RAS

Return Address Stack (RAS)