
# ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

ΕΡΓΑΣΤΗΡΙΟ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

www.cslab.ece.ntua.gr


1 Ιουλίου 2005

## ΠΡΟΗΓΜΕΝΑ ΘΕΜΑΤΑ ΟΡΓΑΝΩΣΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

# 1η Σειρά Ασκήσεων και η Λύση

Θέλουμε να διερευνήσουμε την αποδοτικότητα διαφορετικών μεθόδων πρόβλεψης διακλάδωσης, σχετικά με την επίδοση προγραμμάτων που εκτελούνται σε μία αρχιτεκτονική αγωγού. Στο εξής θεωρούμε ότι διαθέτουμε μία σωλήνωση 5 σταδίων, στην οποία η απόφαση διακλάδωσης πραγματοποιείται στον $2^{ο}$ κύκλο του αγωγού (ID) όταν πρόκειται για εντολή διακλάδωσης χωρίς συνθήκη (`j`, κλπ) και στον $3^{ο}$ κύκλο του αγωγού (EX) όταν πρόκειται για εντολή διακλάδωσης υπό συνθήκη (`beq`, `bne`, κλπ). Θεωρούμε, επίσης ότι υπάρχουν όλα τα δυνατά σχήματα προώθησης.

α) Υποθέτουμε ότι δεν υπάρχει κανενός είδους πρόβλεψη διακλάδωσης (πρέπει να επιλυθεί πρώτα το άλμα πριν συνεχίσει η εκτέλεση της επόμενης εντολής). Πόση καθυστέρηση (σε κύκλους ρολογιού) εισάγουν οι εντολές διακλάδωσης χωρίς συνθήκη και πόση οι εντολές διακλάδωσης υπό συνθήκη;

β) Έστω το παρακάτω τμήμα του κώδικα (ρουτίνα bubblesort):

Θεωρούμε ότι η διεύθυνση του προς ταξινόμηση πίνακα `array` είναι η `100` και το μήκος του πίνακα είναι `Length`. Η τιμή της μεταβλητής `4·i` είναι αρχικά ίση με `Length` και βρίσκεται στον καταχωρητή `$r3`.

| **AO-AE-Label** | **Assembly** | **Σχόλια** |
|---|---|---|
| ***1***-1 loop1: | slt $t1, $zero, $r3 | #while (i > 0) { |
| 2 | beq $t1, $zero, end | |
| ***2***-3 | addi $r5, $zero, 0 | # j = 0; |
| ***3***-4 loop2: | slt $t2, $r5, $r3 | # while (j < i) { |
| 5 | beq $t2, $zero, cont | |
| ***4***-6 | lw $r6, 100($r5) | # temp = array[j]; |
| 7 | lw $r7, 104($r5) | # temp2 = array[j+1]; |
| 8 | slt $t3, $r7, $r6 | # if (temp > temp2) { |
| 9 | beq $t3, $zero, skip | |
| ***5***-10 | sw $r7, 100($r5) | # array[j] = temp2; |
| 11 | sw $r6, 104($r5) | # array[j+1] = temp; |
| | | } |
| ***6***-12 skip: | addi $r5, $r5, 4 | # j++; |
| 13 | j loop2 | # } |
| ***7***-14 cont: | addi $r3, $r3, -4 | # i--; |
| 15 | j loop1 | #} |
| ***8***-16 end: | hlt | |

Ομαδοποιούμε τις εντολές σύμφωνα με την αρίθμηση της στήλης ΑΟ (αριθμός ομάδας). Για κάποιον συγκεκριμένο πίνακα `array`, η σειρά εκτέλεσης των ομάδων εντολών είναι η ακόλουθη:
***1 2 3 4 5 6 3 4 6 3 4 6 3 7 1 2 3 4 5 6 3 4 6 3 7 1 2 3 4 5 6 3 7 1 2 3 7 1 8***

Συμπληρώστε τον ακόλουθο πίνακα με ***N*** όταν δεν πραγματοποιείται άλμα (branch Not taken) και με ***T*** στην περίπτωση που πραγματοποιείται άλμα (branch Taken).

| **ΑΕ** | **Διαδοχικές εκτελέσεις εντολών άλματος** | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 1η | 2η | 3η | 4η | 5η | 6η | 7η | 8η | 9η | 10η |
| 2 | ***N*** | | | | | | | | | |
| 5 | | | | | | | | | | |
| 9 | | | | | | | | | | |
| 13 | | | | | | | | | | |
| 15 | | | | | | | | | | |

Με βάση τον προηγούμενο πίνακα, συμπληρώστε τον πίνακα που ακολουθεί:

| **ΑΕ** | **Αρ.επαναληπτικών Εκτελέσεων** | **Πραγματοποίηση Άλματος (Taken)** | **Όχι Άλμα (Not taken)** | **% Taken** | **% Not taken** |
|---|---|---|---|---|---|
| 2 | | | | | |
| 5 | | | | | |
| 9 | | | | | |
| 13 | | | | | |
| 15 | | | | | |

γ) Δείξτε το διάγραμμα εκτέλεσης των επιμέρους φάσεων για κάθε εντολή (κατά την πρώτη σειρά επαναλήψεων: ***1 2 3 4 5 6*** και ***3 7 1 2***). Υποθέστε ότι υπάρχουν όλα τα δυνατά σχήματα προώθησης (εκτός από τις περιπτώσεις εντολών άλματος), και σχεδιάστε τις θέσεις όπου χρειάζεται να πραγματοποιηθεί προώθηση δεδομένων μεταξύ εντολών. Πόσους κύκλους χρειάζεται για να ολοκληρωθεί η εκτέλεση της ρουτίνας; Πόσοι από τους κύκλους αυτούς σπαταλούνται λόγω εξαρτήσεων δεδομένων και πόσοι λόγω εντολών άλματος (με και χωρίς συνθήκη);

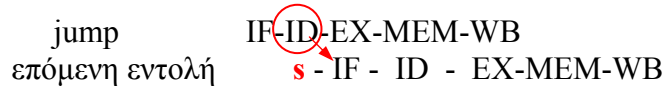
δ) Θεωρείστε μία στατική τεχνική πρόβλεψης διακλάδωσης κατά την οποία ο μεταγλωττιστής υποθέτει πάντοτε ότι οι εντολές άλματος είναι " not taken ".
Ακολουθώντας την ίδια σειρά εκτέλεσης με το ερώτημα (β), πόσους κύκλους καθυστέρησης γλιτώνουμε;

ε) Θεωρείστε εναλλακτικά τη στατική τεχνική πρόβλεψης διακλάδωσης κατά την οποία ο μεταγλωττιστής υποθέτει πάντοτε ότι οι εντολές άλματος είναι "taken".
Ακολουθώντας την ίδια σειρά εκτέλεσης με το ερώτημα (β), πόσους κύκλους καθυστέρησης γλιτώνουμε;

στ) Θεωρείστε μία τεχνική πρόβλεψης διακλάδωσης κατά την οποία ο μεταγλωττιστής «σημαδεύει» κάθε εντολή άλματος ως "taken" ή "not taken" βάσει μίας υποθετικής εκτέλεσης της ρουτίνας, και σύμφωνα με τις στατιστικές που προκύπτουν από αυτήν. Αν, για παράδειγμα, κατά την θεωρητική εκτέλεση η εντολή #2 βρεθεί ότι στο 65% των περιπτώσεων είναι " not taken", θεωρείται καθ' όλη τη διάρκεια της πραγματικής εκτέλεσης ότι μετά την εντολή δεν θα πραγματοποιηθεί άλμα, οπότε συνεχίζει η εκτέλεση των εντολών που την ακολουθούν, πριν ακόμα ολοκληρωθεί η εκτέλεση της εντολή άλματος.
Σύμφωνα με τις στατιστικές του ερωτήματος (β) «σημαδέψτε» κάθε μία από τις 5 εντολές άλματος ως "taken" ή "not taken".
Ακολουθώντας την ίδια σειρά εκτέλεσης με το ερώτημα (β), πόσους κύκλους καθυστέρησης γλιτώνουμε;

# Λύση

α) Με βάση την υπόθεση ότι δεν υπάρχει κανενός είδους πρόβλεψη διακλάδωσης, μία ακολουθία 2 εντολών εκ των οποίων η πρώτη είναι εντολή άλματος θα εκτελείται ως εξής:




Στο στάδιο ID αποφασίζεται ποια είναι η επόμενη εντολή που πρέπει να εκτελεστεί. Από τον επόμενο κύκλο ρολογιού μπορεί να αρχίσει η εκτέλεση της εντολής αυτής. Επομένως, εισάγεται 1 κύκλος καθυστέρησης.


Ομοίως, εισάγονται 2 κύκλοι καθυστέρησης.

β) Για τη συμπλήρωση της πρώτης γραμμής του πίνακα 1, αναζητούμε τις εμφανίσεις της ομάδας AO-***1*** (όπου εμπεριέχεται η εντολή άλματος 2) και ελέγχουμε αν πραγματοποιείται άλμα στη διεύθυνση end ή όχι. Δηλαδή ουσιαστικά ελέγχουμε αν μετά την ομάδα εντολών ***1*** εκτελείται η AO-***2*** (branch Not Taken - ***N***) ή γίνεται άλμα στην AO-***8*** (branch Taken - ***T***). Ομοίως για τις υπόλοιπες εντολές άλματος.

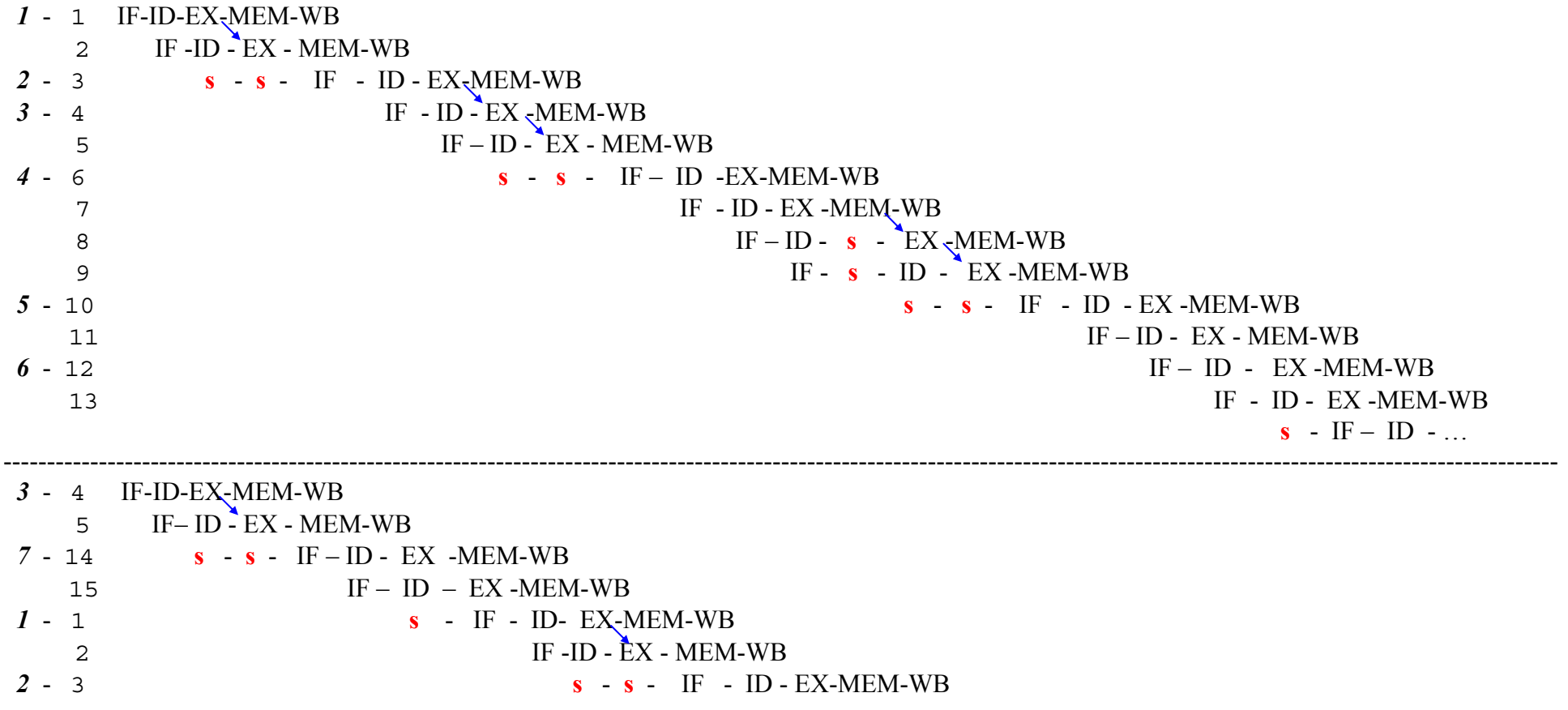
| **AE** | **Διαδοχικές εκτελέσεις εντολών άλματος** | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 1η | 2η | 3η | 4η | 5η | 6η | 7η | 8η | 9η | 10η |
| 2 | ***N*** | ***N*** | ***N*** | ***N*** | ***T*** | | | | | |
| 5 | ***N*** | ***N*** | ***N*** | ***T*** | ***N*** | ***N*** | ***T*** | ***N*** | ***T*** | ***T*** |
| 10 | ***N*** | ***T*** | ***T*** | ***N*** | ***T*** | ***N*** | | | | |
| 14 | ***T*** | ***T*** | ***T*** | ***T*** | ***T*** | ***T*** | | | | |
| 16 | ***T*** | ***T*** | ***T*** | ***T*** | | | | | | |

Με βάση τον προηγούμενο πίνακα, προκύπτουν τα αριθμητικά αποτελέσματα που ακολουθούν:

| **AE** | **Αρ.επαναληπτικών Εκτελέσεων** | **Πραγματοποίηση Άλματος (Taken)** | **Όχι Άλμα (Not taken)** | **% Taken** | **% Not taken** |
|---|---|---|---|---|---|
| 2 | 5 | 1 | 4 | 20% | 80% |
| 5 | 10 | 4 | 6 | 40% | 60% |
| 10 | 6 | 3 | 3 | 50% | 50% |
| 14 | 6 | 6 | 0 | 100% | 0% |
| 16 | 4 | 4 | 0 | 100% | 0% |

γ) Παρακάτω φαίνεται το διάγραμμα εκτέλεσης κατά την πρώτη σειρά επαναλήψεων των εντολών, δηλαδή για την ακολουθία ***1 2 3 4 5 6*** και την ***3 7 1 2***. Οι ακολουθίες αυτές εμπεριέχουν όλες τις ομάδες εντολών. Έτσι, βασιζόμενοι στο διάγραμμα εκτέλεσής τους γίνεται σαφής η καταμέτρηση των κύκλων ρολογιού ολόκληρης της ρουτίνας.

**ΑΟ - ΑΕ**




Όπως φαίνεται και στο αναλυτικό διάγραμμα, δεδομένου ότι δεν υπάρχει κανενός είδους πρόβλεψη διακλάδωσης, ύστερα από κάθε εντολή άλματος υπό συνθήκη (ΑΕ 2, 5 και 10), εισάγεται καθυστέρηση δύο κύκλων (επεξηγήθηκε στο ερώτημα α). Ύστερα από κάθε εντολή άλματος χωρίς συνθήκη (ΑΕ 14 και 16), εισάγεται καθυστέρηση ενός κύκλου.

Τέλος, θεωρούμε ότι υπάρχουν όλα τα δυνατά σχήματα προώθησης, οπότε σημειώνονται τα σημεία όπου υπάρχει ανάγκη να γίνει χρήση της δυνατότητας αυτής. Όλες οι εξαρτήσεις δεδομένων επιλύονται χωρίς εισαγωγή καθυστέρησης, εκτός από την εξάρτηση μεταξύ των εντολών 7 – 8.

| ΑΟ | Αριθμός επαναληπτικών εκτελέσεων | Αριθμός εντολών που εκτελούνται | Αριθμ. stalls λόγω εντολών άλματος | Αριθμ. stalls λόγω εξάρτησης δεδομένων | Συνολικός αριθμ. κύκλων |
|---|---|---|---|---|---|
| ***1*** | 5 | 2 | 2 | - | $5 \cdot (2+2) = 20$ |
| ***2*** | 4 | 1 | - | - | $4 \cdot 1 = 4$ |
| ***3*** | 10 | 2 | 2 | - | $10 \cdot (2+2) = 40$ |
| ***4*** | 6 | 4 | 2 | 1 | $6 \cdot (4+2+1) = 42$ |
| ***5*** | 3 | 2 | - | - | $3 \cdot 2 = 6$ |
| ***6*** | 6 | 2 | 1 | - | $6 \cdot (2+1) = 18$ |
| *7* | 4 | 2 | 1 | - | $4 \cdot (2+1) = 12$ |
| **Σύνολο** | | | $(5+10+6)\cdot 2+(6+4)\cdot 1 = 42 + 10 = 52$ | $6 \cdot 1 = 6$ | 142 |

Όπως προκύπτει από τον παραπάνω πίνακα, η ρουτίνα για να ολοκληρωθεί χρειάζεται:

142 + 4 = 146 κύκλους ρολογιού.

Προσθέτουμε στο σύνολο των εντολών που εκτελούνται μαζί με τις αντίστοιχες καθυστερήσεις που εισάγονται, επιπλέον 4 κύκλους για την αρχικοποίηση του pipeline. Τέλος, παραλείψαμε στην καταμέτρηση την εντολή `17`, επειδή πρόκειται για ψευδοεντολή που δηλώνει ουσιαστικά το τέλος της ρουτίνας.

Από το σύνολο των κύκλων που χρειάζονται για την ολοκλήρωση της ρουτίνας, δαπανώνται 52 κύκλοι λόγω stalls μετά από εντολές άλματος και 6 κύκλοι λόγω εξαρτήσεων δεδομένων.

δ)Υποθέτοντας ότι όλες οι εντολές διακλάδωσης υπό συνθήκη θεωρούνται εξ' ορισμού "not taken", προαποφασίζουμε ότι μετά από κάθε εντολή διακλάδωσης υπό συνθήκη θα εισαχθεί στο pipeline η αμέσως επόμενη εντολή της ακολουθίας, πριν ακόμα γίνει επίλυση του άλματος στο στάδιο EX. (Οι εντολές διακλάδωσης χωρίς συνθήκη δεν θα μπορούσαν να θεωρηθούν "not taken", αφού πάντοτε πραγματοποιούν άλμα).
Οι κύκλοι που γλιτώνουμε από κάθε εντολή είναι:
(ποσοστό επιτυχίας απόφασης Not Taken)×(αρ. επαναλήψεων εντολής)×(αρ. stalls που ακολουθούν)

`2` → Not Taken 80% → $80\% \cdot 5 \cdot 2 = 8$ κύκλοι

`5` → Not Taken 60% → $60\% \cdot 10 \cdot 2 = 12$ κύκλοι

`10` → Not Taken 50% → $50\% \cdot 6 \cdot 2 = 6$ κύκλοι

Σύνολο: 8+12+6 = 26 κύκλοι

ε)Υποθέτοντας ότι όλες οι εντολές διακλάδωσης υπό συνθήκη θεωρούνται εξ' ορισμού "taken", προαποφασίζουμε ότι μετά από κάθε εντολή διακλάδωσης υπό συνθήκη θα εισαχθεί στο pipeline η εντολή-στόχος του άλματος, πριν ακόμα γίνει επίλυση του άλματος στο στάδιο ID (εντολές άλματος χωρίς συνθήκη) ή στο στάδιο EX (εντολές άλματος με συνθήκη). Η διεύθυνση της εντολής-στόχου του άλματος είναι γνωστή εκ των προτέρων στον μεταγλωττιστή και επομένως μπορεί να επιτύχει κατάλληλη αναδιάταξη των εντολών. Εναλλακτικά, το υλικό μπορεί να διαθέτει κατάλληλα σχήματα προώθησης σε προ-σημαδεμένα τμήματα του κώδικα.

Οι κύκλοι που γλιτώνουμε από κάθε εντολή είναι:
(ποσοστό επιτυχίας απόφασης Taken)×(αρ. επαναλήψεων εντολής)×(αρ. stalls που ακολουθούν)

`2` → Taken 20% → $20\% \cdot 5 \cdot 2 = 2$ κύκλοι

`5` → Taken 40% → $40\% \cdot 10 \cdot 2 = 8$ κύκλοι

`10` → Taken 50% → 50% · 6 · 2 = 6 κύκλοι
`14` → Taken 100% → 100% · 6 · 1 = 6 κύκλοι
`16` → Taken 100% → 100% · 4 · 1 =4 κύκλοι

Σύνολο: 2+8+6+6+4 = 26 κύκλοι

στ) Με βάση τα ποσοστά του πίνακα 2 στο ερώτημα (β), αποφασίζουμε ότι θα έχουμε τα εξής «σημάδια» για κάθε εντολή:

`2` → Not Taken
`5` → Not Taken
`10` → Not Taken
`14` → Taken
`16` → Taken

Επομένως κάθε φορά που εκτελείται κάθε μία από τις παραπάνω εντολές θα έχει προαποφασιστεί ποια εντολή θα εισαχθεί στο pipeline, πριν ακόμα γίνει η επίλυση του άλματος στο στάδιο ID (εντολές άλματος χωρίς συνθήκη) ή στο στάδιο EX (εντολές άλματος με συνθήκη). Αν το «σημάδι» δηλώνει ότι πρόκειται για εντολή Not Taken, εισάγεται στο pipeline κανονικά, χωρίς καθυστέρηση, η αμέσως επόμενη εντολή. Αν το «σημάδι» δηλώνει ότι πρόκειται για εντολή Taken, εισάγεται στο pipeline χωρίς καθυστέρηση η επόμενη εντολή που έχει προδιαγραφεί από τον compiler κατά το «σημάδεμα» των εντολών. Τα ποσοστά επιτυχίας της πρωθύστερης απόφασης φαίνονται στον πίνακα 2 του ερωτήματος (β).
Οι κύκλοι που γλιτώνουμε από κάθε εντολή είναι:

(ποσοστό επιτυχίας απόφασης) × (αρ. επαναλήψεων εντολής) × (αρ. stalls που ακολουθούν)

`2` → Not Taken 80% → 80% · 5 · 2 = 8 κύκλοι
`5` → Not Taken 60% → 60% · 10 · 2 = 12 κύκλοι
`10` → Not Taken 50% → 50% · 6 · 2 = 6 κύκλοι
`14` → Taken 100% → 100% · 6 · 1 = 6 κύκλοι
`16` → Taken 100% → 100% · 4 · 1 =4 κύκλοι

Σύνολο: 8+12+6+6+4 = 36 κύκλοι