Πανεπιστήμιο Θεσσαλίας
Τμήμα Πληροφορικής

Οργάνωση Η/Υ

Ενότητα 7η: Ιεραρχία Μνήμης

Άσκηση 1:

Έστω ότι στην εκτέλεση ενός προγράμματος έχουμε διαδοχικές προσπελάσεις των παρακάτω διευθύνσεων μνήμης:

```
1: 0x48c01010      7: 0x48c01020      13: 0x2841101c
2: 0x48c01014      8: 0x28401016      14: 0x48c0104c
3: 0x48c01018      9: 0x48c01024      15: 0x48c01010
4: 0x28401014     10: 0x48c01010      16: 0x48c01014
5: 0x48c0101c     11: 0x48c01014      17: 0x48c01018
6: 0x2841101a     12: 0x48c01048      18: 0x28401018
```

Αν έχουμε σύστημα με ένα επίπεδο κοινής κρυφής μνήμης εντολών και δεδομένων μεγέθους 64KB, χωρισμένης σε πλαίσια των 16 bytes, που αρχικά είναι όλα άκυρα, βρείτε τον αριθμό αποτυχιών στις παραπάνω προσπελάσεις στην κρυφή μνήμη, όταν η οργάνωση της κρυφής μνήμης είναι: (α) άμεσης απεικόνισης, (β) 2-τρόπων συνόλου συσχέτισης, (γ) 4-τρόπων συνόλου συσχέτισης, και (δ) πλήρους συσχέτισης.
Υποθέστε ότι σε περίπτωση ανάγκης αντικατάστασης στις περιπτώσεις (β)-(δ), εφαρμόζεται ο αλγόριθμος LRU.

Απάντηση:

Εφ' όσον το μέγεθος του πλαισίου είναι 16 bytes, το λιγότερο σημαντικό δεκαεξαδικό ψηφίο από τις διευθύνσεις που μας δίνονται χρησιμοποιείται για την προσπέλαση του byte μέσα στο πλαίσιο.
Στην περίπτωση (α) της άμεσης απεικόνισης, η κρυφή μνήμη αποτελείται από 4K πλαίσια, το οποίο σημαίνει ότι χρειαζόμαστε 12 bits, δηλαδή τα 3 επόμενα δεκαεξαδικά ψηφία, για την εύρεση της διεύθυνσης πλαισίου κρυφής μνήμης, όπου πρέπει να προσπελαστεί το δεδομένο. Τα 16 σημαντικότερα bits, δηλαδή τα 4 πιο σημαντικά δεκαεξαδικά ψηφία, είναι αποθηκευμένα στο πλαίσιο στο πεδίο ετικέτας.
Στην περίπτωση (β) οργάνωσης 2-τρόπων συνόλου συσχέτισης, η κρυφή μνήμη αποτελείται από 2K σύνολα των 2 πλαισίων. Αυτό σημαίνει ότι χρειαζόμαστε 11 bits για την εύρεση της διεύθυνσης συνόλου κρυφής μνήμης, στο οποίο πρέπει να αναζητηθεί το πλαίσιο που περιέχει τη συγκεκριμένη διεύθυνση. Τα 17 σημαντικότερα bits είναι αποθηκευμένα στο πεδίο ετικέτας του πλαισίου.
Στην περίπτωση (γ) οργάνωσης 4-τρόπων συνόλου συσχέτισης, η κρυφή μνήμη αποτελείται από 1K σύνολα των 4 πλαισίων. Στην περίπτωση αυτή χρειαζόμαστε 10 bits για την εύρεση της διεύθυνσης συνόλου κρυφής μνήμης, στο οποίο βρίσκεται πιθανά το πλαίσιο όπου θα προσπελαστεί το δεδομένο. Τα 18 σημαντικότερα bits είναι αποθηκευμένα στο πεδίο ετικέτας του πλαισίου.
Τέλος, στην περίπτωση (δ) της οργάνωσης πλήρους συσχέτισης, η κρυφή μνήμη αποτελείται από 4K πλαίσια, στα οποία αναζητείται το πλαίσιο που περιέχει τη διεύθυνση του δεδομένου μας. Και τα 28 bits μετά τα 4 λιγότερο σημαντικά αποθηκεύονται στο πλαίσιο στο πεδίο ετικέτας και χρησιμοποιούνται σαν κλειδί για συσχετιστική αναζήτηση.

Από τη διαδοχή προσπελάσεων που μας δίνεται, μπορούμε να υποθέσουμε ότι οι προσπελάσεις στις διευθύνσεις 0x48c01010 και κάτω – που είναι και οι περισσότερες – είναι προσπε-

λάσεις εντολών, οι οποίες επαναλαμβάνονται από κάποιο σημείο και μετά (ένδειξη δομής βρόχου). Οι υπόλοιπες θα είναι προσπελάσεις δεδομένων, οι οποίες γίνονται σε δύο διαφορετικές περιοχές μνήμης, στην περιοχή διευθύνσεων 0x2840XXXX και στην περιοχή διευθύνσεων 0x2841XXXX. Παρατηρούμε δε ότι οι περισσότερες προσπελάσεις γίνονται σε διευθύνσεις της μορφής 0xXXXX101X, οι οποίες όλες αντιστοιχούν στο ίδιο πλαίσιο για την κρυφή μνήμη άμεσης απεικόνισης, και στο ίδιο σύνολο για τις κρυφές μνήμες 2- και 4-τρόπων συνόλου συσχέτισης.

Για όλες τις περιπτώσεις οργάνωσης της κρυφής μνήμης, σχηματίζουμε τον παρακάτω πίνακα προσπέλασης, όπου με 0 συμβολίζουμε την αποτυχία, και με 1 συμβολίζουμε την επιτυχία στην προσπέλαση της κρυφής μνήμης μας. Σε παρένθεση έχουμε βάλει τις ετικέτες κάθε προσπέλασης, γραμμένες σε δεκαεξαδική βάση.

α/α	**Διεύθυνση**	**Οργάνωση**			
		(α)	(β)	(γ)	(δ)
1	0x48c01010	0 (0x48c0)	0 (0x09180)	0 (0x12300)	0 (0x48c0101)
2	0x48c01014	1 (0x48c0)	1 (0x09180)	1 (0x12300)	1 (0x48c0101)
3	0x48c01018	1 (0x48c0)	1 (0x09180)	1 (0x12300)	1 (0x48c0101)
4	0x28401014	0 (0x2840)	0 (0x05080)	0 (0x0a100)	0 (0x2840101)
5	0x48c0101c	0 (0x48c0)	1 (0x09180)	1 (0x12300)	1 (0x48c0101)
6	0x2841101a	0 (0x2841)	0 (0x05082)	0 (0x0a104)	0 (0x2841101)
7	0x48c01020	0 (0x48c0)	0 (0x09180)	0 (0x12300)	0 (0x48c0102)
8	0x28401016	0 (0x2840)	0 (0x05080)	1 (0x0a100)	1 (0x2840101)
9	0x48c01024	1 (0x48c0)	1 (0x09180)	1 (0x12300)	1 (0x48c0102)
10	0x48c01010	0 (0x48c0)	0 (0x09180)	1 (0x12300)	1 (0x48c0101)
11	0x48c01014	1 (0x48c0)	1 (0x09180)	1 (0x12300)	1 (0x48c0101)
12	0x48c01048	0 (0x48c0)	0 (0x09180)	0 (0x12300)	0 (0x48c0104)
13	0x2841101c	0 (0x2841)	0 (0x05082)	1 (0x0a104)	1 (0x2841101)
14	0x48c0104c	1 (0x48c0)	1 (0x09180)	1 (0x12300)	1 (0x48c0104)
15	0x48c01010	0 (0x48c0)	1 (0x09180)	1 (0x12300)	1 (0x48c0101)
16	0x48c01014	1 (0x48c0)	1 (0x09180)	1 (0x12300)	1 (0x48c0101)
17	0x48c01018	1 (0x48c0)	1 (0x09180)	1 (0x12300)	1 (0x48c0101)
18	0x28401018	0 (0x2840)	0 (0x05080)	1 (0x0a100)	1 (0x2840101)

Έτσι, η πρώτη διεύθυνση προσπέλασης οδηγεί σε όλες τις περιπτώσεις σε αποτυχία, εφ' όσον όλα τα πλαίσια έχουν τιμή 0 στο ψηφίο εγκυρότητας. Το μπλοκ δεδομένων που λαμβάνεται από το επόμενο επίπεδο της ιεραρχίας μνήμης τοποθετείται στο πλαίσιο με διεύθυνση 0x101 για την περίπτωση (α), στο σύνολο με διεύθυνση 0x101 για τις περιπτώσεις (β) και (γ), και σε οποιοδήποτε πλαίσιο της περίπτωσης (δ)[1].
Η δεύτερη και η τρίτη διεύθυνση βρίσκονται στο πλαίσιο που έχουμε ήδη στη μνήμη, κι έτσι η προσπέλαση έχει πάντα επιτυχία[2].
Η τέταρτη διεύθυνση είναι σε περιοχή της μνήμης που δεν έχουμε ξανασυναντήσει, και η προσπέλασή της οδηγεί σε αποτυχία για όλες τις περιπτώσεις. Στην περίπτωση (α), το μπλοκ δεδομένων που λαμβάνεται από το επόμενο επίπεδο ιεραρχίας μνήμης αντικαθιστά το προηγούμενο πλαίσιο, εφ' όσον η διεύθυνση πλαισίου είναι η 0x101, που είναι η ίδια με αυτή των προηγούμενων προσπελάσεων. Στις υπόλοιπες περιπτώσεις, το νέο μπλοκ τοποθετείται σε ένα από τα ελεύθερα πλαίσια. Ειδικά για τις περιπτώσεις (β) και (γ), το μπλοκ αυτό τοποθε-

[1] Ας σημειωθεί ότι ο δεκαεξαδικός αριθμός 0x101 που αναφέρεται ως διεύθυνση προκύπτει από σύμπτυξη 12 bits στην (α), από σύμπτυξη 11 bits στην (β) και από σύμπτυξη 10 bits στην (γ) περίπτωση. Αν η πρώτη διεύθυνση προσπέλασης ήταν 0x4…f010, τότε οι αντίστοιχες διευθύνσεις πλαισίου ή συνόλου θα ήταν 0xf01, 0x701 και 0x301, αντί της ενιαίας 0x101 που σκόπιμα φροντίσαμε να προκύψει. Παρόμοια για τις ετικέτες, που προκύπτουν από σύμπτυξη 16, 17, 18 και 28 bits ανά περίπτωση.
[2] Υποθέτουμε ότι δεν έχουμε διακοπή και επανεκκίνηση του προγράμματος από κάποιον εξωγενή παράγοντα, όπως για παράδειγμα το λειτουργικό σύστημα.

τείται στο σύνολο που προσπελάσαμε προηγουμένως, το οποίο ακόμα δεν είχε γεμίσει, ώστε να χρειαστεί κάποια αντικατάσταση. Στην περίπτωση (β), το σύνολο αυτό γεμίζει με την παρούσα προσπέλαση, δηλαδή τα ψηφία εγκυρότητας και των δύο πλαισίων που περιέχει έχουν τώρα τιμή 1.
Η πέμπτη διεύθυνση βρίσκεται σε πλαίσιο που υπάρχει στη μνήμη για τις περιπτώσεις (β)-(δ), γι' αυτό κι έχουμε επιτυχία για τις περιπτώσεις αυτές. Στην περίπτωση της άμεσης απεικόνισης όμως, το πλαίσιο 0x101 περιέχει τώρα το μπλοκ 0x2840101, που δεν περιλαμβάνει τη διεύθυνση που προσπελαύνουμε. Έτσι, έχουμε αποτυχία, και αναγκαζόμαστε να επαναφέρουμε από το επόμενο επίπεδο ιεραρχίας μνήμης το μπλοκ 0x48c0101.
Η επόμενη διεύθυνση βρίσκεται πάλι σε περιοχή της μνήμης που δεν έχουμε ξανασυναντήσει, κι επομένως έχουμε αποτυχία για όλες τις περιπτώσεις οργάνωσης της κρυφής μνήμης. Ειδικά για τις περιπτώσεις (α)-(γ), παρατηρούμε τα εξής: Στην άμεση απεικόνιση, για άλλη μια φορά η προσπέλαση γίνεται στο πλαίσιο 0x101, οπότε αντικαθιστάμε το περιεχόμενό του με το μπλοκ 0x2841101. Στην οργάνωση 2-τρόπων συνόλου συσχέτισης, το σύνολο 0x101 είναι γεμάτο. Έτσι, ακολουθώντας τον αλγόριθμο LRU, κι εφ' όσον το πλαίσιο που είχαμε φέρει στην προσπέλαση 1 χρησιμοποιήθηκε ξανά στην προσπέλαση 5, αναγκαζόμαστε να αντικαταστήσουμε με το μπλοκ των πιο πάνω διευθύνσεων το πλαίσιο που φέραμε στην προσπέλαση 4. Τέλος, στην περίπτωση (γ), φέρνουμε το συγκεκριμένο μπλοκ χωρίς ανάγκη κάποιας αντικατάστασης, αφού το σύνολο 0x101 έχει ακόμα 2 ελεύθερα πλαίσια.
Στη συνέχεια προσπελαύνουμε μια διεύθυνση που αντιστοιχεί σε μπλοκ διευθύνσεων που δεν έχουμε ξανασυναντήσει, οπότε έχουμε για άλλη μια φορά αποτυχία για όλες τις περιπτώσεις. Η διεύθυνση πλαισίου για την περίπτωση (α) και συνόλου για τις περιπτώσεις (β) και (γ) είναι η 0x102.
Η προσπέλαση 8 αποτυγχάνει για άλλη μια φορά στην περίπτωση (α), επειδή γίνεται στο πλαίσιο 0x101, αλλά σε περιοχή διευθύνσεων διαφορετική από αυτή που προσπελάσαμε την τελευταία φορά που πέσαμε στο ίδιο πλαίσιο, γεγονός που συνέβη στην προσπέλαση 6. Στην περίπτωση (β) έχουμε πάλι αποτυχία, επειδή κανένα από τα δύο πλαίσια που περιέχει το σύνολο 0x101 δεν αντιστοιχεί στη διεύθυνση που προσπελαύνουμε. Αυτή τη φορά θα αντικαταστήσουμε το πλαίσιο που περιέχει το μπλοκ 0x48c0101, επειδή το άλλο χρησιμοποιήθηκε πιο πρόσφατα από αυτό. Στις περιπτώσεις (γ) και (δ) έχουμε επιτυχία.
Από τις υπόλοιπες προσπελάσεις, οι 9, 11, 14, 16 και 17 έχουν επιτυχία σε όλες τις περιπτώσεις, ενώ η προσπέλαση 12 γίνεται σε μπλοκ διευθύνσεων που δεν συναντήθηκε νωρίτερα, κι επομένως έχει αποτυχία για όλες τις περιπτώσεις οργάνωσης.
Οι περιπτώσεις 10, 13 και 18 παρουσιάζουν τα ίδια χαρακτηριστικά με την προσπέλαση 8. Για την περίπτωση (β) αντικαθιστάμε το πλαίσιο με το μπλοκ 0x2841101 κατά την προσπέλαση 10, το πλαίσιο με το μπλοκ 0x2840101 κατά την προσπέλαση 13, και το πλαίσιο με το μπλοκ 0x2841101 κατά την τελευταία προσπέλαση.
Τέλος, η προσπέλαση 15 είναι παρόμοια με την 5, στην οποία μόνο η περίπτωση (α) έχει αποτυχία.

Συνολικά, έχουμε 11 αποτυχίες στην περίπτωση άμεσης απεικόνισης, 9 στην περίπτωση οργάνωσης 2-τρόπων συνόλου συσχέτισης, και 5 στις άλλες δύο περιπτώσεις.
Παρατηρούμε ότι οι αποτυχίες στην οργάνωση 4-τρόπων συνόλου συσχέτισης είναι οι ίδιες με εκείνες της οργάνωσης πλήρους συσχέτισης, γεγονός που οφείλεται στο ότι κανένα σύνολο της περίπτωσης (γ) δε χρησιμοποίησε παραπάνω από 3 από τα 4 πλαίσια που περιέχει.

Άσκηση 2:

Θέλουμε να εκτελέσουμε τον παρακάτω βρόχο C σε μια ΜΕΔ MIPS μερικά επικαλυπτόμενων εντολών με μηχανισμό παροχέτευσης και διακλαδώσεις στατικής πρόβλεψης με έναν κύκλο καθυστέρησης που εκτελούνται στη φάση αποκωδικοποίησης:

```
for (i = 0; i < 1024; i++)
      if (a[i] == 0) a[i] = b[i] + c[i] + d[i];
```

Οι πίνακες ακεραίων a, b, c και d βρίσκονται σε διαδοχικές περιοχές στη μνήμη. Η ΜΕΔ έχει διαχωρισμένη κρυφή μνήμη εντολών και δεδομένων, με 100% επιτυχία στην προσπέλαση της πρώτης. Η δεύτερη έχει μέγεθος 16Kbytes και είναι άμεσης απεικόνισης με πλαίσια των 8 bytes. Η επιτυχημένη προσπέλαση σ' αυτήν γίνεται σε 1 κύκλο μηχανής, ενώ κάθε αποτυχία απαιτεί 6 κύκλους για τη μεταφορά από το επόμενο επίπεδο ιεραρχίας, όπου έχουμε 100% επιτυχία. Τα πλαίσια της κρυφής μνήμης δεδομένων είναι αρχικά άκυρα. Το άλμα στο σώμα του βρόχου εκτελείται στο 40% των επαναλήψεων.

Α. Γράψτε τον παραπάνω βρόχο σε συμβολική γλώσσα, και υπολογίστε το μέσο χρόνο ολοκλήρωσης διαδοχικών επαναλήψεων. Μπορείτε με κατάλληλη αναδιάταξη των εντολών να ελαχιστοποιήσετε τις καθυστερήσεις στην εκτέλεση του βρόχου;

Β. Ξεδιπλώστε το σώμα του βρόχου 2 φορές, αναδιατάξτε τον κώδικα, εάν αυτό το θεωρήσετε απαραίτητο, ώστε να μειώσετε ακόμα περισσότερο τις καθυστερήσεις στην εκτέλεσή του, και υπολογίστε ξανά τον παραπάνω χρόνο.

Γ. Η αρχιτεκτονική MIPS υποστηρίζει την εντολή:

```
pref 0, x($rs)
```

όπου x σταθερά μετατόπισης, για εκ των προτέρων προσκόμιση στην κρυφή μνήμη του μπλοκ που περιέχει τη διεύθυνση x($rs), χωρίς να παρεμποδίζει άλλες επιτυχείς προσπελάσεις από τη ΜΕΔ. Το 0 είναι ένας κωδικός της εντολής που δε μας απασχολεί γι' αυτή την άσκηση.
Με τη χρήση της εντολής pref μειώστε στο ελάχιστο δυνατό τις αποτυχίες στην προσπέλαση της κρυφής μνήμης δεδομένων, και επαναλάβετε τον υπολογισμό του μέσου χρόνου ολοκλήρωσης διαδοχικών επαναλήψεων.

Απάντηση:

Εφ' όσον έχουμε διαχωρισμένη κρυφή μνήμη εντολών και δεδομένων, με την πρώτη να έχει 100% επιτυχία προσπέλασης, θα μας απασχολήσουν μόνο οι προσπελάσεις δεδομένων, αυτές δηλαδή που αφορούν τους πίνακες a, b, c και d. Υποθέτουμε ότι ο δείκτης i δεν αποθηκεύεται στη μνήμη, αλλά διατηρεί την τιμή του σε κάποιον καταχωρητή γενικού σκοπού.
Δεδομένου του μεγέθους της κρυφής μνήμης δεδομένων που είναι 16K bytes και της οργάνωσής της, του συνολικού μεγέθους των τεσσάρων πινάκων που είναι 16K bytes, και του γεγονότος ότι οι πίνακες είναι αποθηκευμένοι σε διαδοχικές περιοχές της μνήμης, δε θα έχουμε περιπτώσεις απεικονίσεων διαφορετικών μπλοκ δεδομένων πάνω στο ίδιο πλαίσιο της κρυφής μνήμης. Έτσι, όλες οι αποτυχίες στην προσπέλαση της κρυφής μνήμης είναι αποτυχίες της πρώτης προσπέλασης (cold misses), εμφανίζονται δηλαδή σε προσπελάσεις σε άκυρα πλαίσια, και δεν οφείλονται σε καμία περίπτωση σε απεικόνιση κάποιας διεύθυνσης σε πλαίσιο που είναι κατειλημμένο από κάποιο ξένο μπλοκ διευθύνσεων.
Ο μηχανισμός παροχέτευσης στη ΜΕΔ μας εξασφαλίζει ότι εξαρτήσεις από δεδομένα αντιμετωπίζονται χωρίς πάγωμα της ΜΕΔ, εκτός από περιπτώσεις δεδομένων που διαβάζονται από τη μνήμη. Επιπλέον, η παροχέτευση δεν καλύπτει πλήρως τις εξαρτήσεις προς εντολές διακλαδώσεων, στις οποίες παρατηρούμε απώλεια ενός κύκλου μηχανής (δύο κύκλων αν η προηγούμενη εντολή είναι ανάγνωση από τη μνήμη). Πιο συγκεκριμένα, αυτό συμβαίνει επειδή υποθέτουμε ότι οι εντολές διακλάδωσης εκτελούνται στη φάση αποκωδικοποίησης της ΜΕΔ, ώστε η αποτυχία στην πρόβλεψη να καλύπτεται πλήρως από τον κύκλο καθυστέρησης. Έτσι όμως, μια πιθανή εξάρτηση από την προηγούμενη εντολή αναγκάζει την εντολή διακλάδωσης να περιμένει την εκτέλεση της εντολής αυτής, παγώνοντας τη ΜΕΔ για έναν κύκλο μηχανής. Ακόμα δε περισσότερο αν η εντολή αυτή είναι εντολή φόρτωσης, οπότε η εντολή διακλάδωσης πρέπει να περιμένει και την προσπέλαση της μνήμης, παγώνοντας δηλαδή τη ΜΕΔ συνολικά για δύο κύκλους μηχανής. Αν δεν υποθέταμε εκτέλεση των διακλαδώσεων στη φάση αποκωδικοποίησης, δε θα είχαμε τέτοιο πάγωμα, όμως θα είχαμε απώλεια ενός κύκλου σε κάθε αποτυχία πρόβλεψης.
Ο ζητούμενος μέσος χρόνος ολοκλήρωσης διαδοχικών επαναλήψεων θα είναι ίσος με το μέσο αριθμό των εντολών ανά επανάληψη, αυξημένος κατά το μέσο αριθμό των εντολών φόρτωσης από τη μνήμη που ακολουθούνται από εξαρτημένη εντολή, το μέσο αριθμό των εντο-

λών διακλάδωσης που είναι εξαρτημένες από την προηγούμενη εντολή, το μέσο αριθμό χαμένων κύκλων λόγω αποτυχημένης πρόβλεψης διακλαδώσεων, και το μέσο αριθμό των χαμένων κύκλων εξαιτίας αποτυχιών στην κρυφή μνήμη. Ο μέσος αριθμός εντολών μας δίνει ουσιαστικά το μέσο χρόνο ιδανικής επικάλυψης, ενώ οι προσαυξήσεις μας δίνουν το μέσο πρόσθετο χρόνο που η ΜΕΔ επιβαρύνεται ανά εντολή, για κάθε περίπτωση μη ιδανικής επικάλυψης.

Α. Μια πιθανή μετάφραση του βρόχου που μας δίνεται είναι η ακόλουθη, όπου υποθέτουμε ότι η αρχική διεύθυνση του πίνακα a βρίσκεται στον καταχωρητή $4:

```
        addi  $5, $4, 4096
Loop:   lw    $8, 0($4)
        bne   $8, $0, X
        nop
        lw    $8, 4096($4)
        lw    $9, 8192($4)
        addu  $8, $8, $9
        lw    $9, 12288($4)
        addu  $8, $8, $9
        sw    $8, 0($4)
X:      addi  $4, $4, 4
        slt   $11, $4, $5
        bne   $11, $0, Loop
        nop
```

Καθώς αυξάνουμε τη διεύθυνση που περιέχει ο $4 κατά 4 σε κάθε επανάληψη του βρόχου, η παράσταση 0($4) μας δίνει τη διεύθυνση του στοιχείου a[i], η παράσταση 4096($4) μας δίνει τη διεύθυνση του στοιχείου b[i], η παράσταση 8192($4) μας δίνει τη διεύθυνση του στοιχείου c[i], ενώ η παράσταση 12288($4) μας δίνει τη διεύθυνση του στοιχείου d[i]. Στον καταχωρητή $5 αποθηκεύουμε τη διεύθυνση τερματισμού του πίνακα a που είναι η επόμενη της διεύθυνσης του τελευταίου στοιχείου του, και η οποία συμπίπτει με την αρχική διεύθυνση του b.

Παρατηρούμε τα εξής:

1. Στην αρχή του σώματος του βρόχου έχουμε μια εξάρτηση από δεδομένα που οδηγεί σε απώλεια δύο κύκλων μηχανής στην εκτέλεση κάθε επανάληψης, και η οποία είναι η εξάρτηση της εντολής bne από την προηγούμενη lw που διαβάζει το a[i]. Σύμφωνα με όσα είπαμε πιο πάνω, αν η διακλάδωση εκτελείται στη φάση εκτέλεσης, η απώλεια είναι μόνο ενός κύκλου μηχανής.
2. Αν το άλμα στο σώμα του βρόχου δεν εκτελείται, έχουμε ακόμα δύο χαμένους κύκλους λόγω εξαρτήσεων από τις εντολές lw που διαβάζουν τα c[i] και d[i]. Η ανάγνωση του b[i] δεν οδηγεί σε απώλεια κανενός κύκλου, επειδή μεταξύ της εντολής ανάγνωσης και της εντολής που χρησιμοποιεί το b[i] μεσολαβεί μία άλλη εντολή.
3. Από τις δύο εντολές διακλάδωσης, καμία δεν περιέχει κάποια χρήσιμη εντολή στη θέση καθυστέρησης, παρά μόνο τη μηδενική εντολή nop. Μπορούμε να δούμε ότι χωρίς αναδιάταξη των εντολών, δεν υπάρχει κάποια χρήσιμη εντολή για να τοποθετηθεί σε θέση καθυστέρησης, είτε επειδή η επόμενη μη μηδενική εντολή λογικά εκτελείται μόνο στη μια κατεύθυνση της διακλάδωσης, είτε επειδή βρισκόμαστε στο τέλος του σώματος του βρόχου και δεν έχουμε άλλη διαθέσιμη εντολή.
4. Στο τέλος του σώματος του βρόχου έχουμε ακόμα μια εξάρτηση από δεδομένα που οδηγεί σε απώλεια ενός κύκλου μηχανής, και συγκεκριμένα την εξάρτηση της εντολής bne από την προηγούμενη εντολή slt. Αν η διακλάδωση εκτελείται στη φάση εκτέλεσης, αυτή η απώλεια δεν υπάρχει.
5. Όσο αφορά τις αποτυχίες στην κρυφή μνήμη, λόγω του μεγέθους του πλαισίου που περιλαμβάνει 2 λέξεις, θα εξεταστούν ανά ζεύγος επαναλήψεων. Έτσι, αν το άλμα στο εσωτερικό του βρόχου δεν εκτελείται έστω και μια φορά για ένα ζεύγος διαδοχικών επαναλήψεων, εμφανίζονται 4 αποτυχίες – μία για κάθε πίνακα. Στην περίπτωση αυτή, οι αποτυχίες αυτές θα δημιουργούν καθυστέρηση 4×6 = 24 κύκλων. Εάν όμως το άλμα εκτελεστεί και τις δύο φορές, δε θα έχουμε ανάγνωση του αντίστοιχου πλαισίου των πινάκων b,

c και d. Στην περίπτωση αυτή, η απώλεια περιορίζεται στους 6 κύκλους μηχανής – αυτούς που αντιστοιχούν στην ανάγνωση ενός πλαισίου του πίνακα a.

Σύμφωνα με τα παραπάνω, ο μέσος χρόνος ολοκλήρωσης διαδοχικών επαναλήψεων θα είναι 13×60% + 7×40% = 10,6 κύκλοι μηχανής λόγω του αριθμού εντολών που εκτελούνται ανά επανάληψη, συν 4×60% + 2×40% = 3,2 κύκλοι μηχανής λόγω των εξαρτήσεων από εντολές lw, συν 1 κύκλος μηχανής λόγω της εξάρτησης της εντολής bne από την slt, συν 1/2×24×84% + 1/2×6×16% = 10,56 κύκλοι μηχανής[3] λόγω των αποτυχιών στην κρυφή μνήμη. Συνολικά επομένως ο ζητούμενος χρόνος θα είναι 25,36 κύκλοι μηχανής. Λόγω της εκτέλεσης των διακλαδώσεων στη φάση αποκωδικοποίησης και του ενός κύκλου καθυστέρησης, δεν έχουμε επιβάρυνση από τις αποτυχίες στην πρόβλεψη διακλαδώσεων[4].

Θα προσπαθήσουμε τώρα να βρούμε μια αναδιάταξη των εντολών στο σώμα του βρόχου, ώστε (i) να εξαλείψουμε ή να μειώσουμε τις καθυστερήσεις εξαρτήσεων από εντολές lw ή εξαρτήσεων εντολών διακλάδωσης από προηγούμενες εντολές, (ii) να τοποθετήσουμε κάποιες χρήσιμες εντολές στις θέσεις καθυστέρησης των διακλαδώσεων, και (iii) να μειώσουμε τις απώλειες κύκλων σε αποτυχίες της κρυφής μνήμης. Ας σημειωθεί ότι αν και οι εντολές nop εκτελούνται κανονικά, γενικά οι κύκλοι εκτέλεσής τους μετρούνται ως χαμένοι κύκλοι, οπότε είναι σκόπιμο να αποφεύγουμε τη χρήση τέτοιων εντολών όσο είναι δυνατό.
Κατ' αρχήν είναι εύκολο να παρατηρήσουμε ότι οι απώλειες κύκλων από τις αποτυχίες της κρυφής μνήμης δεν αντιμετωπίζονται με απλή αναδιάταξη των εντολών, επειδή είναι όλες αποτυχίες πρώτης προσπέλασης των πινάκων.
Από τις υπόλοιπες απώλειες κύκλων, μπορούμε να εξαλείψουμε όσες οφείλονται στις εξαρτήσεις από τις δύο εντολές lw που διαβάζουν τα c[i] και d[i] με απλή αναδιάταξη εντολών στο τμήμα μεταξύ της εντολής διακλάδωσης και της διεύθυνσης προορισμού X, και κατάλληλη μετονομασία καταχωρητών.
Επίσης μπορούμε να χρησιμοποιήσουμε τις δύο εντολές που ακολουθούν για να καλύψουμε δύο ακόμα χαμένους κύκλους. Επιλέγουμε αυθαίρετα να καλύψουμε με αυτές τις θέσεις καθυστέρησης των εντολών διακλάδωσης. Αυτό έχει σαν αποτέλεσμα να εξαλείψουμε και την καθυστέρηση από την εξάρτηση της δεύτερης διακλάδωσης από την εντολή slt, όταν το άλμα της πρώτης δεν εκτελείται.
Μας μένουν δύο ακόμα κύκλοι, αυτοί που προκύπτουν από την εξάρτηση της πρώτης διακλάδωσης από την πρώτη εντολή φόρτωσης, οι οποίοι δε μπορούν να καλυφτούν χωρίς να ξεδιπλώσουμε το βρόχο.

Η νέα μορφή του βρόχου είναι η ακόλουθη:

```
      addi  $5, $4, 4092
Loop: lw    $8, 0($4)
      bne   $8, $0, X
      slt   $11, $4, $5
      lw    $8, 4096($4)
      lw    $9, 8192($4)
      lw    $10, 12288($4)
      addu  $8, $8, $9
      addu  $8, $8, $10
      sw    $8, 0($4)
X:    bne   $11, $0, Loop
      addi  $4, $4, 4
```

[3] Δύο διαδοχικές εκτελέσεις του άλματος της διακλάδωσης στο σώμα του βρόχου συμβαίνουν με πιθανότητα 40%×40% = 16%, οπότε έστω και μία μη εκτέλεση – ή ισοδύναμα το πολύ μία εκτέλεση – συμβαίνει με πιθανότητα 84%. Για να πάρουμε την επιβάρυνση ανά επανάληψη στην περίπτωση ζεύγους επαναλήψεων, πολλαπλασιάζουμε επί ½ την επιβάρυνση ανά ζεύγος.

[4] Μπορείτε να βρείτε πόση θα ήταν αυτή η επιβάρυνση, αν η εκτέλεση των διακλαδώσεων γινόταν στην επόμενη φάση. Τότε ο ένας κύκλος καθυστέρησης δε θα έφτανε για να καλύψει το κόστος της αποτυχίας, που θα ήταν 1 κύκλος μηχανής λόγω της εντολής που λανθασμένα ανακλήθηκε από τη μνήμη. Από την άλλη μεριά, βέβαια, σε τέτοια περίπτωση δε θα είχαμε πρόσθετα παγώματα σε εξαρτήσεις διακλαδώσεων από την προηγούμενη εντολή.

Παρατηρούμε ότι (i) χρησιμοποιήσαμε έναν ακόμα καταχωρητή στο σώμα του βρόχου, όπου τοποθετούμε την τιμή του d[i], και (ii) λόγω της μεταφοράς της εντολής σύγκρισης πριν την αύξηση του $4 κατά 4, αναγκαστήκαμε να θέσουμε στον $5 μειωμένη τιμή από την προηγούμενη κατά 4, αυτή δηλαδή που αντιστοιχεί στη διεύθυνση του τελευταίου στοιχείου του a.

Ο μέσος χρόνος ολοκλήρωσης διαδοχικών επαναλήψεων του βρόχου είναι τώρα: 11×60% + 5×40% = 8,6 κύκλοι μηχανής από το μέσο αριθμό εντολών του σώματος του βρόχου, συν 2 κύκλοι από την εξάρτηση από την πρώτη εντολή lw, συν 1×40% = 0,4 κύκλοι μηχανής από την εξάρτηση της δεύτερης διακλάδωσης από την εντολή slt, η οποία οδηγεί σε απώλεια ενός κύκλου μόνο όταν η πρώτη διακλάδωση εκτελεί άλμα, συν 10,56 κύκλοι – όπως και πριν – για την αποτυχία της κρυφής μνήμης. Συνολικά είναι ίσος με 21,56 κύκλοι μηχανής.

Β. Το ξεδίπλωμα του βρόχου αποσκοπεί (i) στη μείωση του αριθμού συνολικών εντολών που εκτελούνται στο βρόχο, (ii) στην εξάλειψη τυχόν εξαρτήσεων που δεν αντιμετωπίζονται επιτυχώς στην απλή επανάληψη, και (iii) στην καλύτερη εν γένει εκμετάλλευση της ΜΕΔ με αναδιάταξη των εντολών για εκτέλεση χωρίς καθυστερήσεις.

Το αποτέλεσμα του ξεδιπλώματος στο συγκεκριμένο βρόχο είναι το εξής:

```
      addi  $5, $4, 4088
Loop: lw    $8, 0($4)
      bne   $8, $0, X1
      lw    $11, 4($4)
      lw    $8, 4096($4)
      lw    $9, 8192($4)
      lw    $10, 12288($4)
      addu  $8, $8, $9
      addu  $8, $8, $10
      sw    $8, 0($4)
X1:   bne   $11, $0, X2
      slt   $11, $4, $5
      lw    $8, 4100($4)
      lw    $9, 8196($4)
      lw    $10, 12292($4)
      addu  $8, $8, $9
      addu  $8, $8, $10
      sw    $8, 4($4)
X2:   bne   $11, $0, Loop
      addi  $4, $4, 8
```

Στον κώδικα αυτόν εφαρμόσαμε παρόμοια αναδιάταξη εντολών με αυτήν που εφαρμόσαμε και προηγουμένως.
Με το ξεδίπλωμα επιτυγχάνουμε τη μείωση του αριθμού συνολικών εντολών, επειδή τόσο η ενημέρωση του καταχωρητή $4 που παίζει το ρόλο του δείκτη στο βρόχο, όσο και η εκτέλεση της εντολής διακλάδωσης που τελειώνει το βρόχο, γίνονται μόνο στο ½ των αρχικών επαναλήψεων.
Με την αναδιάταξη που εφαρμόσαμε, ισχύουν οι παρατηρήσεις που κάναμε και προηγουμένως. Επιπλέον, βελτιώσαμε τη δεύτερη διακλάδωση, επειδή η ανάγνωση από τη μνήμη από την οποία αυτή εξαρτάται, μεταφέρθηκε αρκετές εντολές νωρίτερα, κι έτσι η τιμή που αυτή χρειάζεται είναι διαθέσιμη όταν εκτελείται, όμως μόνο σε περίπτωση που η πρώτη διακλάδωση δεν εκτελέσει άλμα. Ειδικότερα, όταν το άλμα στην πρώτη διακλάδωση δεν εκτελείται, η δεύτερη διακλάδωση δεν εμφανίζει απώλεια κύκλων, επειδή βρίσκεται σε αρκετή απόσταση από την αντίστοιχη φόρτωση. Όταν όμως το άλμα εκτελείται, η τιμή του a[i+1] που διαβάζουμε δεν έχει ακόμα εμφανιστεί στη ΜΕΔ, κι έτσι θα έχουμε την προηγούμενη απώλεια των δύο κύκλων μηχανής.
Περισσότερες βελτιώσεις δε μπορούμε να πετύχουμε, επειδή δεν έχουμε διαθέσιμες εντολές να παρεμβάλουμε μεταξύ της πρώτης εντολής lw και της εντολής bne που ακολουθεί, ώστε να αποφύγουμε τα παγώματα λόγω της μεταξύ τους εξάρτησης, ούτε αμέσως πριν από τη δεύτερη εντολή bne, ώστε να αποφύγουμε τα παγώματα στη δεύτερη διακλάδωση όταν η

πρώτη εκτελεί το άλμα της, αλλά ούτε και αμέσως πριν από την τελευταία διακλάδωση, ώστε να αποφύγουμε το πάγωμα που είχαμε και προηγουμένως λόγω της εξάρτησής της από την εντολή slt που δημιουργεί κίνδυνο όταν η δεύτερη διακλάδωση εκτελεί το άλμα της.

Ο μέσος χρόνος ολοκλήρωσης διαδοχικών επαναλήψεων του νέου βρόχου θα είναι τώρα 19×36% + 13×48% + 7×16% = 14,2 κύκλοι μηχανής[5] από τον αριθμό εντολών στο σώμα του βρόχου, συν 2 κύκλοι μηχανής από την εξάρτηση από την πρώτη εντολή lw, συν 2×40% = 0,8 κύκλοι από την εξάρτηση της δεύτερης διακλάδωσης από τη δεύτερη εντολή lw, η οποία οδηγεί σε απώλεια δύο κύκλων μόνο όταν η πρώτη διακλάδωση εκτελεί άλμα, συν 1×40% = 0,4 κύκλοι από την εξάρτηση της τρίτης διακλάδωσης από την εντολή slt, η οποία οδηγεί σε απώλεια ενός κύκλου μόνο όταν η δεύτερη διακλάδωση εκτελεί άλμα, συν 2×10,56 = 21,12 κύκλοι από τις αποτυχίες προσπέλασης κρυφής μνήμης. Συνολικά ο χρόνος είναι 38,52 κύκλοι ανά επανάληψη του νέου βρόχου, που αντιστοιχεί σε 19,26 κύκλους μηχανής ανά επανάληψη του αρχικού βρόχου.

Γ. Η εντολή pref προσκομίζει ένα μπλοκ δεδομένων στην κρυφή μνήμη εκ των προτέρων, πριν δηλαδή χρειαστούν τα δεδομένα που αυτό περιέχει. Η εντολή αυτή δεν παγώνει τη ΜΕΔ με την εκτέλεσή της, αλλά ολοκληρώνεται με την ενεργοποίηση της μεταφοράς του μπλοκ προς την κρυφή μνήμη, επιτρέποντας τις επόμενες εντολές να προχωρήσουν κανονικά.
Επιπλέον, η μεταφορά του μπλοκ προς την κρυφή μνήμη δεν εμποδίζει άλλες προσπελάσεις να πραγματοποιηθούν, αν αυτές έχουν επιτυχία στην κρυφή μνήμη. Αν όμως εμφανιστεί κάποια αποτυχία ή αν προσπαθήσουμε να εκτελέσουμε νέα εντολή pref καθώς περιμένουμε ένα μπλοκ από το επόμενο επίπεδο της ιεραρχίας μνήμης, θα πρέπει να παγώσουμε τη ΜΕΔ, μέχρι η πρώτη μεταφορά να ολοκληρωθεί[6].
Εφ' όσον θέλουμε δύο επαναλήψεις του αρχικού βρόχου για να εξαντλήσουμε τα δεδομένα ενός πλαισίου, θα χρησιμοποιήσουμε τον ξεδιπλωμένο βρόχο για να παρεμβάλουμε εντολές pref μεταξύ των εντολών του σώματος του βρόχου. Επειδή προσπελαύνουμε 4 διαφορετικούς πίνακες σε κάθε επανάληψη, χρειαζόμαστε 4 εντολές pref, καθεμία από τις οποίες να αναφέρεται σε διεύθυνση που αντιστοιχεί ένα μπλοκ πιο κάτω από τη διεύθυνση που προσπελαύνει κανονικά η παρούσα επανάληψη. Με μέγεθος μπλοκ ίσο με 8 bytes, αρκεί να χρησιμοποιήσουμε μετατοπίσεις αυξημένες κατά 8 από εκείνες των κανονικών προσπελάσεων.
Η προσθήκη των εντολών pref αυξάνει το μήκος του σώματος του βρόχου, δηλαδή το μέσο αριθμό εντολών που εκτελούνται σε κάθε επανάληψη, αλλά εξαλείφει ή μειώνει τις καθυστερήσεις από αποτυχίες στην κρυφή μνήμη. Αυτό συμβαίνει επειδή κάθε εντολή pref φέρνει ένα μπλοκ δεδομένων στην κρυφή μνήμη, χωρίς να χρειάζεται να παγώσουμε τη ΜΕΔ περιμένοντας το μπλοκ, επειδή δεν το χρειαζόμαστε αμέσως.
Μπορούμε να τοποθετήσουμε τις 4 εντολές pref, έτσι ώστε να μειώσουμε ακόμα περισσότερο τις καθυστερήσεις που έχουμε στην εκτέλεση του βρόχου. Όμως, πρέπει να είμαστε προσεκτικοί και να τοποθετήσουμε διαδοχικές εντολές pref σε κατάλληλη απόσταση μεταξύ τους, ώστε η εκτέλεση κάθε επόμενης pref να μην επικαλύπτεται με μια προηγούμενη.
Ένας τρόπος χρησιμοποίησης της εντολής pref είναι ο παρακάτω:

```
        addi  $5, $4, 4088
Loop:   lw    $8, 0($4)
        pref  0, 8($4)
        lw    $11, 4($4)
        slt   $12, $4, $5
        bne   $8, $0, X1
        pref  0, 4104($4)
        lw    $8, 4096($4)
        lw    $9, 8192($4)
        lw    $10, 12288($4)
```

[5] Τα βάρη στους αριθμούς κύκλων προκύπτουν συνδυάζοντας κατάλληλα τις πιθανότητες εκτέλεσης άλματος με όλους τους δυνατούς τρόπους για δύο διαδοχικές επαναλήψεις του αρχικού βρόχου.
[6] Υπάρχουν και κρυφές μνήμες που επιτρέπουν εκκρεμείς προσπελάσεις στο επόμενο επίπεδο της ιεραρχίας μνήμης, και δεν οδηγούν τη ΜΕΔ σε πάγωμα, αν έρθει νέα προσπέλαση όσο εκκρεμεί κάποια άλλη. Τέτοιες μνήμες τις λέμε μνήμες *που δεν αποκλείονται (non-blocking)*.

```
        addu  $8, $8, $9
        addu  $8, $8, $10
        sw    $8, 0($4)
X1:     bne   $11, $0, X2
        pref  0, 8200($4)
        lw    $8, 4100($4)
        lw    $9, 8196($4)
        lw    $10, 12292($4)
        addu  $8, $8, $9
        addu  $8, $8, $10
        sw    $8, 4($4)
X2:     pref  0, 12296($4)
        bne   $12, $0, Loop
        addi  $4, $4, 8
```

Παρατηρούμε ότι με τις εντολές pref προσκομίζουμε δεδομένα στην κρυφή μνήμη, ακόμα κι αν δεν πρόκειται να τα χρησιμοποιήσουμε. Επειδή η πιθανότητα να μην προσπελαστεί ένα μπλοκ δεδομένων των πινάκων b, c και d είναι χαμηλή – και συγκεκριμένα ίση με την πιθανότητα δύο διαδοχικών εκτελέσεων άλματος στο σώμα του βρόχου που είναι 16%, η επιβάρυνση από την άσκοπη προσκόμιση δεδομένων είναι μικρή. Αν η πιθανότητα εκτέλεσης των αλμάτων ήταν μεγαλύτερη, θα ήταν προτιμότερο να μη χρησιμοποιήσουμε την τεχνική της εκ των προτέρων προσκόμισης, αλλά να προσκομίσουμε τα μπλοκ δεδομένων των πινάκων αυτών με τις εντολές lw που τα διαβάζουν. Κάτι τέτοιο θα μας το υποδείκνυε και ο χρόνος ολοκλήρωσης διαδοχικών επαναλήψεων που θα προέκυπτε σε τέτοια περίπτωση.
Αν και με την προσθήκη των νέων εντολών εξαλείφουμε πλήρως τις καθυστερήσεις στις εξαρτήσεις από εντολές φόρτωσης και τις εξαρτήσεις προς εντολές διακλάδωσης, η διαδοχή των 4 εντολών pref δημιουργεί κάποιες καθυστερήσεις λόγω των διαδοχικών προσκομίσεων μπλοκ δεδομένων προς την κρυφή μνήμη. Έτσι, όταν εμφανίζεται κάποια εντολή pref, η ΜΕΔ θα παγώσει, αν τη στιγμή που η εντολή αυτή μπει στη φάση προσπέλασης της μνήμης, πραγματοποιείται κάποια άλλη προσκόμιση μπλοκ προς την κρυφή μνήμη. Αυτό γίνεται περισσότερο αισθητό όταν οι διακλαδώσεις στο σώμα του βρόχου εκτελούν άλμα, οπότε οι εντολές pref εμφανίζονται πολύ κοντά η μία με την άλλη.
Από την άλλη μεριά, αν η προσπέλαση ενός πλαισίου γίνεται από 6 και πάνω κύκλους μηχανής μετά την εντολή pref που το προσκομίζει, έχουμε επιτυχία στην προσπέλασή του. Με τον τρόπο αυτό εξαλείφουμε πλήρως τις καθυστερήσεις από τις αποτυχίες κρυφής μνήμης. Κάτι τέτοιο επιτυγχάνεται στον κώδικά μας, αφού οι εντολές pref αναφέρονται σε δεδομένα της επόμενης επανάληψης και σε κάθε περίπτωση εκτελούνται αρκετές εντολές για να καλυφθεί ο ελάχιστος αριθμός των 6 κύκλων μηχανής. Οι μόνες αποτυχίες που παραμένουν είναι αυτές της πρώτης επανάληψης, οι οποίες δε μας απασχολούν.

Ο χρόνος ολοκλήρωσης διαδοχικών επαναλήψεων καθορίζεται στις περισσότερες περιπτώσεις από την απόσταση μεταξύ διαδοχικών εντολών pref. Για όλες τις περιπτώσεις εκτέλεσης μιας επανάληψης, η απόσταση μεταξύ της πρώτης και της δεύτερης pref είναι 4 εντολές. Σύμφωνα με τα παραπάνω, αυτή δε μπορεί να αντιστοιχεί σε λιγότερους από 6 κύκλους μηχανής. Παρόμοια, η απόσταση μεταξύ της τελευταίας και της πρώτης pref αντιστοιχεί αναγκαστικά σε 6 κύκλους μηχανής. Για τις υπόλοιπες διαδοχές pref έχουμε:

1. Αν και οι δύο διακλαδώσεις στο σώμα του βρόχου εκτελούν άλμα, οι αποστάσεις μεταξύ της δεύτερης και της τρίτης και μεταξύ της τρίτης και της τέταρτης pref είναι μικρές, οπότε αναγκαστικά αντιστοιχούν σε 2×6 = 12 κύκλους μηχανής, με σύνολο 24 κύκλους για όλη την επανάληψη.
2. Αν καμία από τις δύο διακλαδώσεις δεν εκτελεί άλμα, ο χρόνος μεταξύ δεύτερης και τρίτης pref καθορίζεται από τον αριθμό των πραγματικών εντολών που εκτελούνται, και είναι ίσος με 8 κύκλους μηχανής. Παρόμοια, ο χρόνος μεταξύ τρίτης και τέταρτης pref θα είναι 7 κύκλοι μηχανής. Άρα συνολικά θα έχουμε 27 κύκλους για όλη την επανάληψη.
3. Αν η πρώτη διακλάδωση δεν εκτελεί άλμα και η δεύτερη εκτελεί, θα έχουμε 8 και 6 κύκλους για τους αντίστοιχους χρόνους, με σύνολο 26 κύκλους μηχανής.

4. Αν η πρώτη διακλάδωση εκτελεί άλμα και η δεύτερη δεν εκτελεί, θα έχουμε 6 και 7 κύκλους για τους αντίστοιχους χρόνους, με σύνολο 25 κύκλους μηχανής.

Κατά μέσο όρο θα έχουμε 24×16% + 25×24% + 26×24% + 27×36% = 25,8 κύκλους, που αντιστοιχούν σε 12,9 κύκλους της επανάληψης του αρχικού βρόχου.

Ο παραπάνω χρόνος είναι καλύτερος απ' όσους βρήκαμε μέχρι τώρα, επειδή δεν έχουμε πια αποτυχίες στην προσπέλαση της κρυφής μνήμης. Στην πράξη, ο αριθμός 8 είναι μικρός για μέγεθος πλαισίου κρυφής μνήμης. Αν είχαμε ένα πλαίσιο των 16 bytes, θα ξεδιπλώναμε το βρόχο 4 φορές, οπότε η προσθήκη των 4 εντολών pref δε θα είχε εισάγει τις καθυστερήσεις λόγω μικρής απόστασης μεταξύ διαδοχικών pref. Σε μια τέτοια περίπτωση όμως, πιθανό να μην καταφέρναμε να καλύψουμε πλήρως τις καθυστερήσεις από τις εξαρτήσεις από δεδομένα, επειδή θα είχαμε περισσότερες περιπτώσεις εξαρτήσεων να αντιμετωπίσουμε στο σώμα του βρόχου με τον ίδιο όμως αριθμό πρόσθετων εντολών.

Άσκηση 3:

Θεωρήστε μια κρυφή μνήμη πλήρους συσχέτισης μεγέθους 16K bytes, χωρισμένη σε πλαίσια των 16 bytes.
Παρουσιάστε μια υλοποίηση του αλγορίθμου αντικατάστασης LRU, προσθέτοντας σε κάθε πλαίσιο όσα ψηφία θεωρείτε απαραίτητο, ώστε να απεικονίσετε την ιστορία προσπέλασης για την εύρεση του λιγότερο πρόσφατα χρησιμοποιηθέντος πλαισίου.
Ποιος είναι ο ελάχιστος αριθμός συμπληρωματικών ψηφίων που απαιτούνται ανά πλαίσιο; Τι υλικό χρειάζεστε για την υλοποίηση με αυτό τον αριθμό ψηφίων;
Αν έχετε λιγότερα συμπληρωματικά ψηφία από τον ελάχιστο αριθμό ψηφίων που απαιτούνται, σκεφτείτε αν η υλοποίησή σας προσεγγίζει τον αλγόριθμο LRU με ικανοποιητικό τρόπο.

Απάντηση:

Για την απεικόνιση της ιστορίας προσπέλασης σε ένα σύνολο από 1K πλαίσια, μπορούμε να χρησιμοποιήσουμε έναν μετρητή ανά πλαίσιο. Έτσι, με κατάλληλο αλγόριθμο τροποποίησης της τιμής του μετρητή κάθε πλαισίου, αυτό με τη μέγιστη τιμή μετρητή θα είναι το λιγότερο πρόσφατα χρησιμοποιηθέν πλαίσιο που θα αντικατασταθεί στην επόμενη προσπέλαση, αν κάτι τέτοιο χρειαστεί. Ο μετρητής θα πρέπει να αυξάνει την τιμή του σε κάθε προσπέλαση κάποιου άλλου πλαισίου, ενώ το πλαίσιο που προσπελαύνεται θα πρέπει να μηδενίζει την τιμή του μετρητή του, ώστε να αναγνωρίζεται σαν το *πιο* πρόσφατα χρησιμοποιηθέν πλαίσιο, για να αντικατασταθεί το αργότερο δυνατό.

Αν είχαμε την ευχέρεια χρησιμοποίησης απεριόριστου αριθμού ψηφίων ανά μετρητή, ο πιο απλός αλγόριθμος θα ήταν ο πιο κάτω, με c_i την τιμή του μετρητή για το πλαίσιο i:

Αρχικοποίηση:
$c_i = \infty$;
Σε κάθε προσπέλαση του πλαισίου i:
for ($j \neq i$) c_j++;
$c_i = 0$;
Σε κάθε αντικατάσταση:
LRU = i, $c_i = \max_j\{c_j\}$;
προσκόμιση_μπλοκ(LRU);
for ($j \neq i$) c_j++;
$c_i = 0$;

Ο ελάχιστος αριθμός ψηφίων ανά μετρητή για την απεικόνιση της ιστορίας προσπελάσεων των 1K πλαισίων, ώστε κάθε μετρητής να έχει μοναδική τιμή, είναι 10, αφού αυτός είναι ο ελάχιστος αριθμός bits που μπορούν να αναπαραστήσουν τους αριθμούς από 0 έως 1023.
Με αυτόν τον αριθμό ψηφίων, ο αλγόριθμος τροποποιείται ως εξής:

Αρχικοποίηση:
$c_i = 1023;$
Σε κάθε προσπέλαση του πλαισίου i:
for ($j \neq i$) if ($c_j < c_i$) c_j++;
$c_i = 0;$
Σε κάθε αντικατάσταση:
LRU = i, $c_i = 1023;$
προσκόμιση_μπλοκ(LRU);
for ($j \neq i$) c_j++;
$c_i = 0;$

Η αύξηση κατά 1 και στις δύο περιπτώσεις εξασφαλίζει ότι οι μετρητές έχουν μοναδική τιμή, αφού αυξάνονται κατά 1 όλοι όσοι δεν προσπελαύνονται στην πρώτη και όλοι όσοι έχουν μικρότερη τιμή στη δεύτερη περίπτωση, ενώ μόνο ένας μετρητής μηδενίζεται κάθε φορά. Το πλαίσιο LRU στη δεύτερη περίπτωση έχει πάντα τιμή μετρητή 1023.
Για την υλοποίηση του αλγορίθμου LRU με 10 συμπληρωματικά bits ανά πλαίσιο, χρειαζόμαστε ένα κύκλωμα σύγκρισης του μετρητή κάθε πλαισίου με συγκεκριμένη τιμή, και επακόλουθης αύξησης της τιμής του ανάλογα με το αποτέλεσμα της σύγκρισης. Η εύρεση ενός πλαισίου με τιμή μετρητή ίση με 1023 γίνεται εύκολα με συσχετιστική αναζήτηση με κλειδί την τιμή του μετρητή.

Στην περίπτωση που δε διαθέτουμε 10 bits ανά πλαίσιο για την υλοποίηση των μετρητών, μπορούμε να χρησιμοποιήσουμε λιγότερα ψηφία με τον ίδιο αλγόριθμο. Στην περίπτωση αυτή ο αλγόριθμός μας θα ακολουθεί τη συμπεριφορά του LRU για λιγότερο από 1024 πλαίσια κάθε φορά. Αν για παράδειγμα διαθέτουμε 5 ψηφία, ο αλγόριθμος θα διατηρεί τα 31 πιο πρόσφατα χρησιμοποιηθέντα πλαίσια, και δε θα αυξάνει την τιμή του μετρητή πέρα από τη μέγιστη τιμή 31. Για αντικατάσταση, θα επιλέγει εξ ίσου από τα υπόλοιπα 993, άσχετα αν μερικά από αυτά χρησιμοποιήθηκαν πιο πρόσφατα από άλλα. Μια τεχνική τυχαίας επιλογής μας εξασφαλίζει τη χρήση όλων των πλαισίων.
Προφανώς η συμπεριφορά αυτή προσεγγίζει απλά τη συμπεριφορά του LRU για τα 1024 πλαίσια της κρυφής μνήμης, και είναι θέμα εφαρμογής το πόσο καλά τον προσεγγίζει. Μ' άλλα λόγια, αν η εφαρμογή μπορεί να επαναπροσπελάσει στο μέλλον ένα από τα πλαίσια που έχει προσπελάσει, το οποίο όμως έχει τιμή μετρητή ίση με 31, ο αλγόριθμος θα είναι κατώτερος του LRU. Διαφορετικά η επίδοση της κρυφής μνήμης, τουλάχιστον όσο αφορά την αντικατάσταση πλαισίων θα είναι ταυτόσημη με αυτή της κρυφής μνήμης που χρησιμοποιεί τον ακριβή αλγόριθμο LRU.

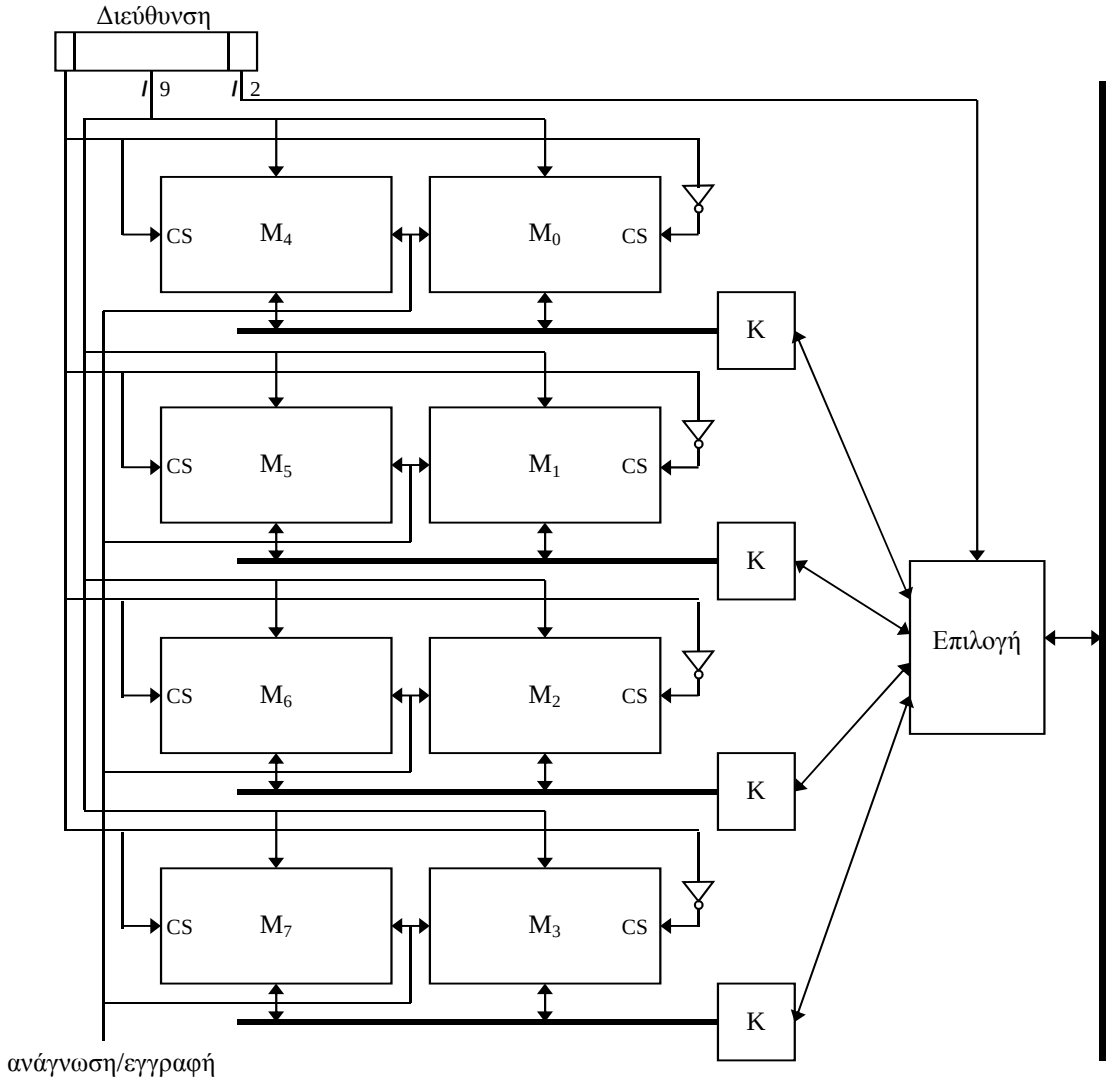
Άσκηση 4:

Να σχεδιάσετε σύστημα μνήμης με διαφύλλωση χαμηλής τάξης, για υπολογιστή με χώρο διευθύνσεων των 12 bits. Το διαθέσιμο υλικό μνήμης είναι τμήματα μεγέθους 512x8 bits. Πόσων δρόμων είναι η μέγιστη διαφύλλωση που μπορείτε να έχετε; Υποθέστε ότι το μέγεθος λέξης είναι 8 bits, και ότι κάθε προσπέλαση γίνεται για μία λέξη.

Απάντηση:

Ο χώρος διευθύνσεων που έχουμε διευθυνσιοδοτεί μέχρι $2^{12} = 4096$ λέξεις στη μνήμη. Εφ' όσον το υλικό μνήμης που διαθέτουμε είναι τμήματα των 512 λέξεων, θα χρειαστούμε 8 τμήματα, αν θέλουμε να καλύψουμε όλο το χώρο διευθύνσεων.
Η μέγιστη διαφύλλωση χαμηλής τάξης που μπορούμε να έχουμε είναι 4096/512 = 8 δρόμων, που επιτυγχάνεται όταν όλα τα τμήματα συμμετέχουν στη διαφύλλωση χαμηλής τάξης. Σε περίπτωση που επιθυμούμε μικρότερη διαφύλλωση, για παράδειγμα 4 δρόμων, θα χρησιμοποιήσουμε διαφύλλωση υψηλής τάξης για κάθε δύο τμήματα. Προφανώς, ο βαθμός διαφύλλωσης χαμηλής τάξης πρέπει να διαιρεί τον αριθμό τμημάτων που χρησιμοποιούμε.

Ένα σύστημα μνήμης με χαμηλή διαφύλλωση 4 δρόμων δίνεται σχηματικά παρακάτω:



Κάθε τμήμα μνήμης διευθυνσιοδοτεί τους ακόλουθους χώρους:

M_0: 0,4,8,…,2044
M_1: 1,5,9,…,2045
M_2: 2,6,10,…,2046
M_3: 3,7,11,…,2047
M_4: 2048,2052,…,4092
M_5: 2049,2053,…,4093
M_6: 2050,2054,…,4094
M_7: 2051,2055,…,4095

Σε κάθε προσπέλαση, το πιο σημαντικό bit της διεύθυνσης χρησιμοποιείται για την επιλογή του τμήματος (CS) υψηλής διαφύλλωσης, ενώ τα 9 επόμενα bits προσπελαύνουν μια θέση μνήμης από τα 4 τμήματα που επιλέχθηκαν. Για ανάγνωση, μεταφέρονται 4 λέξεις στους αντίστοιχους καταχωρητές K, ενώ τα 2 λιγότερα σημαντικά bits της διεύθυνσης επιλέγουν τη σωστή λέξη με τη βοήθεια ενός πολυπλέκτη. Η λέξη αυτή στέλνεται στο αμέσως χαμηλότερο επίπεδο ιεραρχίας μνήμης. Για εγγραφή, μπορούμε να ξεκινήσουμε με τον ίδιο τρόπο και να διαβάσουμε αρχικά 4 λέξεις στους καταχωρητές K, να μεταφέρουμε στη συνέχεια τη λέξη για εγγραφή στον κατάλληλο καταχωρητή, και τέλος να εγγράψουμε τις τιμές και των 4 καταχωρητών πίσω στα 4 τμήματα που επιλέχθηκαν στην πρώτη φάση.

Άσκηση 5:

Θεωρήστε ένα σύστημα ιδεατής μνήμης με τα πιο κάτω χαρακτηριστικά:

- *Χώρος φυσικών διευθύνσεων των 36 bits.*
- *Χώρος ιδεατών διευθύνσεων των 40 bits.*
- *Η ιδεατή μνήμη είναι χωρισμένη σε σελίδες των 16 KB.*

Ποιο είναι το συνολικό μέγεθος του πίνακα σελίδων για κάθε διεργασία σ' αυτό το σύστημα, αν μια διεργασία μπορεί να χρησιμοποιήσει το μέγιστο αριθμό σελίδων, και αν ο πίνακας σελίδων περιλαμβάνει για κάθε σελίδα και 4 bits για εγκυρότητα, προστασία, αλλαγή και χρήση; Υποθέστε ότι ο πίνακας σελίδων δεν αποθηκεύει διευθύνσεις βοηθητικής μνήμης.

Απάντηση:

Εφ' όσον ο χώρος διευθύνσεων ιδεατής μνήμης είναι των 40 bits και το μέγεθος σελίδας είναι 16KB, η ιδεατή μνήμη θα αποτελείται από $2^{40}/16K = 2^{40}/2^{14} = 2^{26}$ σελίδες. Η φυσική μνήμη θα αποτελείται αντίστοιχα από $2^{36}/16K = 2^{36}/2^{14} = 2^{22}$ ενότητες.

Θεωρούμε ότι κάθε διεργασία μπορεί να προσπελάσει το μέγιστο χώρο ιδεατής μνήμης. Επομένως, ο πίνακας σελίδων για κάθε διεργασία θα περιέχει 2^{26} γραμμές, όσες δηλαδή οι σελίδες της ιδεατής μνήμης. Κάθε γραμμή θα έχει 22 + 4 = 26 bits, από τα οποία 22 bits διευθυνσιοδοτούν ενότητες φυσικής μνήμης και τα υπόλοιπα 4 bits είναι τα ψηφία εγκυρότητας, προστασίας, αλλαγής και χρήσης[7]. Συνολικά επομένως χρειαζόμαστε:

$$2^{26} \times 26 \text{ bits} = 208 \text{ MB}$$

κι αυτό εφ' όσον μπορούμε να συμπτύξουμε τις γραμμές του πίνακα σελίδων στο μέγιστο δυνατό βαθμό. Διαφορετικά, θα θέλαμε 4 bytes για κάθε γραμμή, σύνολο 256 MB.

Προφανώς, το συνολικό μέγεθος ενός πίνακα σελίδων είναι πολύ μεγάλο για να βρίσκεται συνεχώς στην κύρια μνήμη. Η απλούστερη λύση γι' αυτό το πρόβλημα είναι βέβαια ο πίνακας σελίδων να είναι κι αυτός μέρος της ιδεατής μνήμης, ώστε να μπορεί να αποθηκεύεται στη βοηθητική μνήμη. Μπορούμε ακόμα να υλοποιήσουμε τον πίνακα σελίδων ιεραρχικά, με ένα βασικό πίνακα μεγέθους 1 σελίδας να διευθυνσιοδοτεί ένα μεγαλύτερο πίνακα, ο οποίος να περιέχει τις διευθύνσεις των ενοτήτων που θέλουμε.
Εναλλακτικά, μπορούμε να περιορίσουμε το χώρο διευθύνσεων κάθε διεργασίας να περιλαμβάνει ένα μέρος μόνο της ιδεατής μνήμης, χρησιμοποιώντας την τεχνική της κατάτμησης, ώστε μια διεργασία να προσπελαύνει ένα τμήμα μόνο της ιδεατής μνήμης. Ένας πίνακας σελίδων μεγέθους 1 σελίδας μπορεί να περιέχει τις απεικονίσεις τμήματος μέχρι 4K σελίδων, θεωρώντας ότι κάθε γραμμή του πίνακα απαιτεί 4 bytes.
Τέλος, για να εξασφαλίσουμε γρήγορη προσπέλαση μνήμης, μπορούμε να χρησιμοποιήσουμε και μια ειδική κρυφή μνήμη, την TLB (Translation-Lookaside Buffer), ώστε να αποθηκεύουμε την απεικόνιση των πιο πρόσφατα χρησιμοποιηθέντων σελίδων σε ενότητες. Έτσι, εάν η απεικόνιση της σελίδας που προσπελαύνουμε βρίσκεται στην TLB, βρίσκουμε την αντίστοιχη ενότητα, χωρίς να απαιτείται καθόλου προσπέλαση του πίνακα σελίδων.

Άσκηση 6:

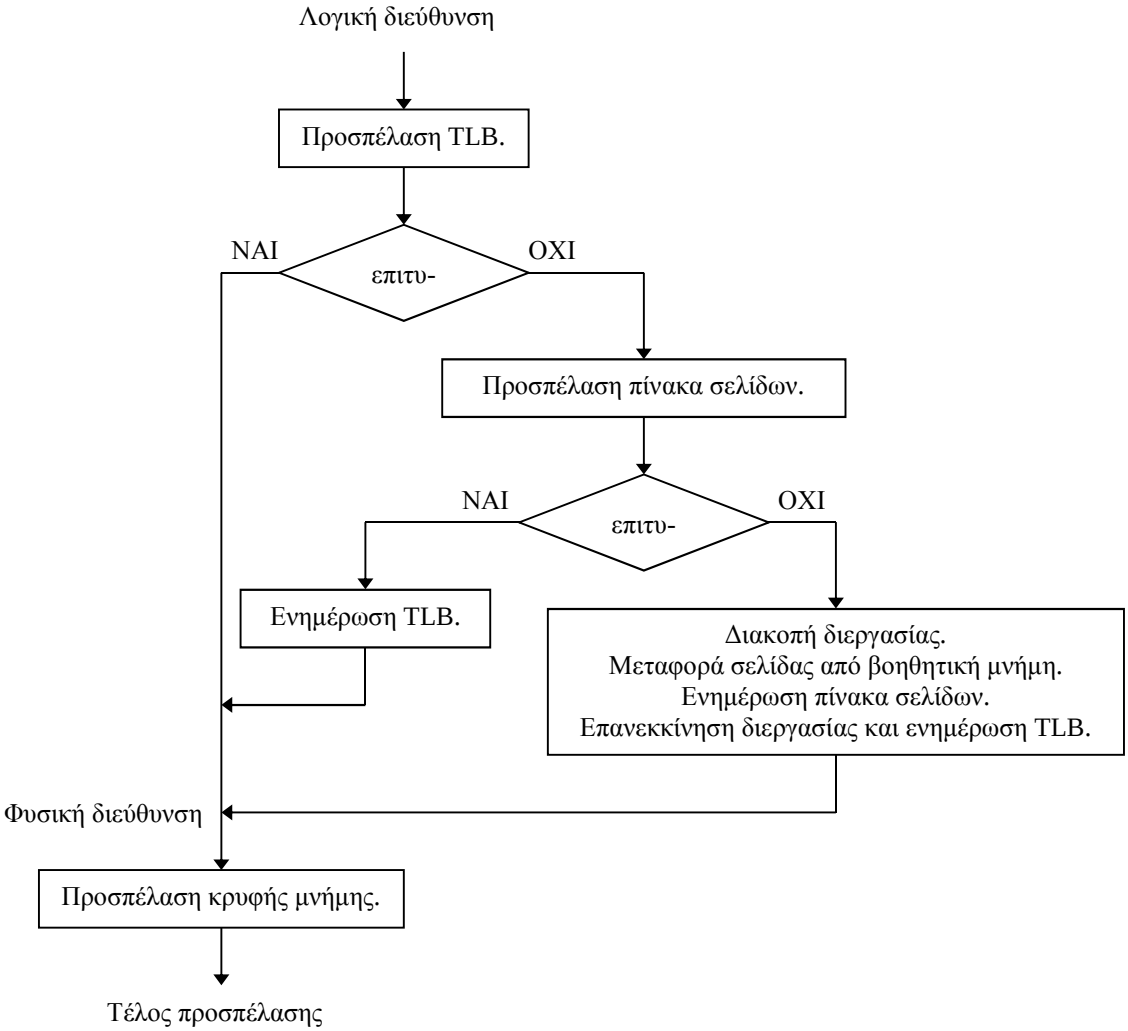
Σε ένα σύστημα μνήμης που περιλαμβάνει τόσο κρυφή όσο και ιδεατή μνήμη με TLB, μια προσπέλαση μνήμης μπορεί να αποτύχει σε 3 διαφορετικά σημεία: Αποτυχία κρυφής μνήμης, αποτυχία στην TLB, και αποτυχία σελίδας.

[7] Το ψηφίο εγκυρότητας δείχνει αν η συγκεκριμένη ενότητα που διευθυνσιοδοτούν τα 22 bits αντιστοιχεί στη σελίδα, το ψηφίο προστασίας δείχνει αν η διεργασία μπορεί να γράψει στη σελίδα, το ψηφίο αλλαγής δείχνει αν η σελίδα έχει γραφτεί από τη διεργασία, ενώ το ψηφίο χρήσης δείχνει αν η σελίδα χρησιμοποιήθηκε πρόσφατα, ώστε να μην αλλαχτεί σε περίπτωση αποτυχίας κάποιας άλλης σελίδας.

Θεωρήστε όλους τους δυνατούς συνδυασμούς αποτυχιών (δηλαδή προσπελάσεις με τουλάχιστον μία από τις παραπάνω αποτυχίες), και εξηγήστε για κάθε συνδυασμό (α) αν είναι εφικτός και (β) αν ναι, πότε μπορεί να συμβεί.

Απάντηση:

Ο μηχανισμός προσπέλασης μιας διεύθυνσης μνήμης ξεκινάει με προσπέλαση της TLB. Αν εκεί βρεθεί η διεύθυνση ενότητας που περιέχει τη ζητούμενη διεύθυνση, μετατρέπουμε τη λογική διεύθυνση στην αντίστοιχη φυσική, και συνεχίζουμε με την προσπέλαση της κρυφής μνήμης[8]. Διαφορετικά, ανατρέχουμε στον πίνακα σελίδων για να βρούμε την ενότητα κύριας μνήμης που θέλουμε, κι αν δεν τη βρούμε ούτε εκεί, έχουμε αποτυχία σελίδας με εμπλοκή του λειτουργικού συστήματος για μεταφορά της αντίστοιχης σελίδας από τη βοηθητική μνήμη.
Σχηματικά ο μηχανισμός προσπέλασης μέχρι την προσπέλαση της κρυφής μνήμης δίνεται παρακάτω. Στον πίνακα που ακολουθεί φαίνονται οι συνδυασμοί αποτυχιών στην προσπέλαση, ενώ δίνονται και τα σχετικά σχόλια.



[8] Πολλές κρυφές μνήμες διευθυνσιοδοτούνται με λογικές διευθύνσεις, ώστε η προσπέλαση να είναι πιο γρήγορη, καθώς στην περίπτωση αυτή δε χρειάζεται να προηγηθεί προσπέλαση της TLB πριν την προσπέλαση της κρυφής μνήμης. Η τεχνική αυτή έχει μεγαλύτερη πολυπλοκότητα από την απλή τεχνική χρήσης φυσικών διευθύνσεων στην κρυφή μνήμη, μια που έτσι η απεικόνιση από την κρυφή μνήμη στο επόμενο επίπεδο μνήμης πρέπει να περάσει από μετάφραση. Επιπλέον, κι εφ' όσον κάθε διεργασία έχει δικό της χώρο ιδεατής μνήμης, υπάρχει και πρόβλημα επικάλυψης των λογικών διευθύνσεων διαφορετικών διεργασιών, που μπορεί να βρίσκονται ταυτόχρονα στην κρυφή μνήμη.

Κρυφή μνήμη	TLB	Σελίδα	Σχόλια
Αποτυχία	Αποτυχία	Αποτυχία	Εφικτός συνδυασμός. Η προσπέλαση της TLB αποτυγχάνει, καθώς η σελίδα δε βρίσκεται στην κύρια μνήμη. Η σελίδα μεταφέρεται από τη βοηθητική μνήμη, και η επακόλουθη προσπέλαση της κρυφής μνήμης αποτυγχάνει, αφού κανένα μπλοκ από τη σελίδα αυτή δεν έχει μεταφερθεί ακόμα στην κρυφή μνήμη. Ο πίνακας σελίδων και η TLB ενημερώνονται κατάλληλα.
Αποτυχία	Αποτυχία	Επιτυχία	Εφικτός συνδυασμός. Η προσπέλαση της TLB αποτυγχάνει, αλλά η σελίδα βρίσκεται στην κύρια μνήμη. Η προσπέλαση της κρυφής μνήμης αποτυγχάνει. Η TLB ενημερώνεται κατάλληλα.
Αποτυχία	Επιτυχία	X	Εφικτός συνδυασμός. Η διεύθυνση ενότητας της κύριας μνήμης λαμβάνεται από την TLB, αλλά η προσπέλαση της κρυφής μνήμης που ακολουθεί είναι αποτυχημένη. Ο πίνακας σελίδων δεν εξετάζεται όταν η TLB έχει επιτυχία.
Επιτυχία	Αποτυχία	Αποτυχία	Αδύνατο. Με τη σελίδα απούσα από την κύρια μνήμη, δεν επιτρέπεται η κρυφή μνήμη να περιέχει έγκυρο μπλοκ από αυτή.
Επιτυχία	Αποτυχία	Επιτυχία	Εφικτός συνδυασμός. Η προσπέλαση της TLB αποτυγχάνει, αλλά η σελίδα βρίσκεται στην κύρια μνήμη. Η προσπέλαση της κρυφής μνήμης είναι επιτυχημένη. Η TLB ενημερώνεται.
Επιτυχία	Επιτυχία	X	Εφικτός συνδυασμός. Η διεύθυνση ενότητας της κύριας μνήμης λαμβάνεται από την TLB, και η προσπέλαση της κρυφής μνήμης είναι επίσης επιτυχημένη. Ο πίνακας σελίδων δεν εξετάζεται όταν η TLB έχει επιτυχία.

Από το διάγραμμα προσπέλασης μπορούμε να παρατηρήσουμε ότι όταν η TLB έχει επιτυχία, δεν ελέγχεται ο πίνακας σελίδων. Έτσι, οι συνδυασμοί που έχουν επιτυχία στην TLB, τοποθετήθηκαν στον πίνακα αποτυχιών με αδιάφορο το αποτέλεσμα προσπέλασης του πίνακα σελίδων.
Αν όμως εξετάζαμε τον πίνακα σελίδων, θα είχαμε οπωσδήποτε επιτυχία, διότι δε μπορεί η TLB να περιέχει απεικόνιση σελίδας που δε βρίσκεται στην κύρια μνήμη. Αυτό συμβαίνει, επειδή κάθε φορά που μια σελίδα διώχνεται από την κύρια μνήμη, η TLB ακυρώνει την αντίστοιχη απεικόνιση, εάν βέβαια την περιέχει.

Άσκηση 7:

Θεωρήστε τον παρακάτω κώδικα συμβολικής γλώσσας MIPS:

```
Loop: lw    $8, 0($4)
      beq   $8, $0, X
      lw    $8, 4($4)
      lw    $9, 8($4)
      addu  $8, $8, $9
X:    addiu $4, $4, 12
      sw    $8, 0($5)
      addiu $5, $5, 4
      bne   $4, $18, Loop
```

Ο κώδικας αυτός εκτελείται σε μια ΜΕΔ MIPS μερικά επικαλυπτόμενων εντολών με μηχανισμό παροχέτευσης και δυναμική πρόβλεψη διακλαδώσεων δύο ψηφίων ιστορίας με μνήμη που παρέχει τη διεύθυνση προορισμού παράλληλα με την πρόβλεψη. Οι διακλαδώσεις εκτελούνται στη φάση εκτέλεσης, αλλά ολοκληρώνονται στη φάση προσπέλασης μνήμης, οπότε και ενημερώνεται ο μηχανισμός πρόβλεψης. Η ΜΕΔ έχει ενοποιημένη κρυφή μνήμη εντολών και δεδομένων με μία θύρα ανάγνωσης/εγγραφής.

Α. Δείξτε πώς ο κώδικας εκτελείται στην ΜΕΔ, δίνοντας και εξηγώντας το διάγραμμα χρονισμού των εκτελούμενων εντολών για πλήρη εκτέλεση των τεσσάρων πρώτων επαναλήψεων του βρόχου. Υποθέστε ότι πριν εμφανιστεί η πρώτη εντολή, η ΜΕΔ είναι άδεια, ενώ δεν υπάρχει καμία αποθηκευμένη πληροφορία στην υπομονάδα πρόβλεψης διακλαδώσεων. Χρησιμοποιήστε τον αλγόριθμο δυναμική πρόβλεψης που μελετήσαμε στο μάθημα, με μηδενικά αρχικά ψηφία ιστορίας. Για τον παραπάνω κώδικα, υποθέστε ότι η διακλάδωση στο σώμα του βρόχου δεν εκτελεί άλμα στην πρώτη, εκτελεί άλμα στη δεύτερη και στην τρίτη και δεν εκτελεί άλμα στην τέταρτη επανάληψη, ενώ η διακλάδωση τερματισμού του βρόχου εκτελεί άλμα και στις τέσσερεις επαναλήψεις. Υποθέστε ότι οι διευθύνσεις των εντολών διακλάδωσης δεν εμφανίζουν σύγκρουση στην προσπέλαση της μνήμης ιστορίας και διευθύνσεων προορισμού.

Β. Αν στην αρχή της εκτέλεσης του παραπάνω κώδικα ο PC περιέχει τιμή 0x1001a0c0, ο καταχωρητής $4 περιέχει τιμή 0x1006a0b0 και ο καταχωρητής $5 περιέχει τιμή 0x1008a0c0, βρείτε την ακολουθία διευθύνσεων που στέλνονται από τη ΜΕΔ στην κρυφή μνήμη για την εκτέλεση των τεσσάρων επαναλήψεων του βρόχου. Αγνοώντας τη μετάφραση των διευθύνσεων από λογικές σε φυσικές, και υποθέτοντας ότι πριν εμφανιστεί η πρώτη εντολή τα πλαίσια της κρυφής μνήμης είναι άκυρα, δείξτε τις ευστοχίες, αστοχίες και αντικαταστάσεις στην προσπέλαση της κρυφής μνήμης για τις διευθύνσεις που βρήκατε, με την υπόθεση ότι η κρυφή μνήμη έχει μέγεθος 128Kbytes με πλαίσια των 16 bytes, συνολοσυσχετιστική οργάνωση δύο δρόμων με ετερόχρονη εγγραφή, προσκόμιση κατά την εγγραφή και αντικατάσταση LRU.

Γ. Αν για μεγάλο πλήθος επαναλήψεων η διακλάδωση στο σώμα του βρόχου έχει πρόβλεψη εκτέλεσης άλματος στο 30%, με ομοιόμορφη επιτυχία για όλες τις προβλέψεις, και εκτελεί άλμα στο 40% των επαναλήψεων, υπολογίστε το μέσο χρόνο ολοκλήρωσης διαδοχικών επαναλήψεων του βρόχου. Χρησιμοποιήστε το διάγραμμα χρονισμού του προηγούμενου ερωτήματος και προσθέστε όποια διαγράμματα σας λείπουν για τον υπολογισμό. Υποθέστε ότι η κρυφή μνήμη ευστοχεί σε ποσοστό 98%, οπότε η προσπέλαση σ' αυτήν γίνεται σε 1 κύκλο μηχανής, διαφορετικά απαιτούνται 6 πρόσθετοι κύκλοι για τη μεταφορά από το επόμενο επίπεδο ιεραρχίας, όπου έχουμε 100% επιτυχία.

Απάντηση:

Α. Για να βρούμε το διάγραμμα χρονισμού για την εκτέλεση του κώδικα, θα πρέπει κατ' αρχήν να εξετάσουμε τις εξαρτήσεις σε αυτόν και να εντοπίσουμε τους κινδύνους.
Επειδή έχουμε μία ενοποιημένη κρυφή μνήμη εντολών και δεδομένων, υπάρχουν δομικές εξαρτήσεις στις προσπελάσεις μνήμης. Από τη στιγμή που η ΜΕΔ είναι επικαλυπτόμενη, ενεργοποιείται ανάκληση εντολής σε κάθε κύκλο μηχανής, επομένως δημιουργούνται δομικοί κίνδυνοι κάθε φορά που μια εντολή φόρτωσης ή αποθήκευσης φτάνει στη φάση προσπέλασης μνήμης (Φ4), με κάποια επόμενη εντολή που βρίσκεται στη φάση ανάκλησης (Φ1). Σε περίπτωση εμφάνισης ενός τέτοιου κινδύνου η μια από τις δύο προσπελάσεις πρέπει να περιμένει την άλλη, κάτι που θα οδηγήσει σε πάγωμα της ΜΕΔ από τη φάση στην οποία εμφανίζεται αυτή η προσπέλαση και προς τα πίσω. Έτσι, δεν έχει νόημα να παγώσει η εντολή φόρτωσης ή αποθήκευσης που βρίσκεται στη Φ4, μια που τότε θα πάγωνε και η εντολή που βρίσκεται στη Φ1, αλλά αντίθετα θα παγώσει η δεύτερη, ώστε η πρώτη να προχωρήσει κανονικά. Αν η εντολή που βρίσκεται στη Φ4 δεν είναι εντολή φόρτωσης ή αποθήκευσης, προφανώς δεν τίθεται θέμα δομικής εξάρτησης στη μνήμη. Στη ΜΕΔ που εξετάζουμε δεν έχουμε άλλου είδους δομικές εξαρτήσεις.

Όσο αφορά τις εξαρτήσεις δεδομένων στο φάκελο καταχωρητών, μπορούμε να τις βρούμε κατασκευάζοντας έναν πίνακα με όλες τις προσπελάσεις καταχωρητών που εμπλέκονται σε τουλάχιστον μία εγγραφή, ως εξής:

Εντολή	Καταχωρητής			
	$4	$5	$8	$9
E1	A		E	
E2			A	
E3	A		E	
E4	A			E
E5			A/E	A
E6	A/E			
E7		A	A	
E8		A/E		
E9	A			

όπου έχουμε αριθμήσει τις εντολές χάριν συντομίας, ενώ με Α και Ε εννοούμε ανάγνωση και εγγραφή αντίστοιχα. Καταχωρητές που δεν γράφονται από καμία εντολή του κώδικα δε χρειάζεται να συμπεριληφθούν, αφού δεν είναι δυνατό να συμμετέχουν σε εξαρτήσεις δεδομένων.

Από τον παραπάνω πίνακα μπορούμε εύκολα να εντοπίσουμε τις εξαρτήσεις δεδομένων. Αν με αστερίσκο σημειώνουμε εντολές επόμενης επανάληψης, οι εξαρτήσεις τύπου ΑΜΕ για τον καταχωρητή $4 είναι: (α) E6 → E9, (β) E6 → E1*, (γ) E6 → E3*, (δ) E6 → E4* και (ε) E6 → E6*, για τον $5 είναι: (στ) E8 → E7* και (ζ) E8 → E8*, για τον $8 είναι: (η) E1 → E2, (θ) E1 → E7, (ι) E3 → E5 και (ια) E5 → E7, και τέλος για τον καταχωρητή $9 είναι η (ιβ) E4 → E5. Οι εξαρτήσεις τύπου ΕΜΑ για τον καταχωρητή $4 είναι: (ιγ) E1 → E6, (ιδ) E3 → E6, (ιε) E4 → E6, (ιστ) E6 → E6 και (ιζ) E9 → E6*, για τον $5 είναι: (ιη) E7 → E8 και (ιθ) E8 → E8, για τον $8 είναι: (κ) E2 → E3, (κα) E2 → E1*, (κβ) E5 → E5 και (κγ) E7 → E1*, ενώ για τον $9 είναι η (κδ) E5 → E4*. Τέλος, οι εξαρτήσεις τύπου ΕΜΕ για τον $4 είναι: (κε) E6 → E6*, για τον $5 είναι: (κστ) E8 → E8*, για τον $8 είναι: (κζ) E1 → E3, (κη) E1 → E1*, (κθ) E3 → E5 και (λ) E5 → E1*, ενώ για τον $9 είναι η (λα) E4 → E4*.

Στις εξαρτήσεις δεδομένων δεν αναφέρουμε εκείνες που καλύπτονται από άλλες εξαρτήσεις, όπως για παράδειγμα την ΑΜΕ εξάρτηση E1 → E5 για τον $8, μια που η E5 δε χρησιμοποιεί ποτέ δεδομένα που παράγει η E1, λόγω της ΕΜΕ εξάρτησης (κζ) που αναγκάζει την E5 να διαβάσει δεδομένα που γράφει η E3. Όμως, πρέπει να είμαστε προσεκτικοί, διότι λόγω της εντολής διακλάδωσης E2 δημιουργούνται δύο ροές ελέγχου στον κώδικα, κι έτσι δε μπορούμε να παραβλέψουμε εξαρτήσεις που ενεργοποιούνται όταν εκτελείται άλμα από την E2 προς την εντολή E6. Έτσι, για παράδειγμα η ΑΜΕ εξάρτηση (θ) από την E1 προς την E7 δεν παραλείπεται, παρόλο που για μη εκτέλεση άλματος της E2 καλύπτεται από άλλες εξαρτήσεις. Αν το άλμα εκτελεστεί, η E7 θα είναι άμεσα εξαρτημένη από την E1, και όχι από την E5 που δεν θα έχει εκτελεστεί.

Έχοντας εντοπίσει όλες τις εξαρτήσεις δεδομένων που πρέπει να τηρούνται κατά την εκτέλεση του κώδικά μας, μπορούμε να προχωρήσουμε περισσότερο και να διαχωρίσουμε τις εξαρτήσεις που αποτελούν κίνδυνο κατά την εκτέλεση σε ΜΕΔ MIPS μερικής επικάλυψης με μηχανισμό παροχέτευσης, καθώς επίσης και τις εξαρτήσεις στις οποίες ο κίνδυνος αποτρέπεται μέσω του μηχανισμού παροχέτευσης, από τις υπόλοιπες.

Γνωρίζουμε ότι οι εξαρτήσεις τύπου ΕΜΑ και ΕΜΕ δεν αποτελούν κίνδυνο στη ΜΕΔ MIPS που διαθέτουμε, αφού πάντα οι εγγραφές επόμενων εντολών συμβαίνουν μετά τις αναγνώσεις και εγγραφές των προηγούμενων. Επιπλέον, δεν χρειάζονται το μηχανισμό παροχέτευσης για την τήρησή τους, μια που δεν μεταφέρουν δεδομένα από κάποια εντολή σε κάποια επόμενη. Από τις εξαρτήσεις τύπου ΑΜΕ κίνδυνο αποτελούν μόνο όσες δημιουργούνται από δεδομένα που φορτώνονται από τη μνήμη, και όταν η εξαρτημένη εντολή είναι η αμέσως επόμενη. Οι εξαρτήσεις που αποτελούν κίνδυνο οδηγούν σε πάγωμα ενός κύκλου κατά την εκτέλεση του

κώδικα, μέχρι τα δεδομένα να προσκομιστούν από τη μνήμη. Οι εξαρτήσεις που αποτελούν κίνδυνο στον κώδικά μας είναι μόνο οι (η) και (ιβ).
Οι εξαρτήσεις τύπου ΑΜΕ που τηρούνται μέσω του μηχανισμού παροχέτευσης είναι όσες αφορούν εντολές που η εξαρτημένη εντολή χρησιμοποιεί τα δεδομένα έναν ή δύο κύκλους μετά από την εντολή που τα παράγει. Οι εξαρτήσεις που αποτελούν κίνδυνο τηρούνται μέσω του μηχανισμού παροχέτευσης μετά το αναγκαστικό πάγωμα που επιφέρει ο κίνδυνος. Επομένως, με πρώτη ματιά οι εξαρτήσεις που αντιμετωπίζονται με παροχέτευση είναι οι (η), (ι), (ια) και (ιβ). Όμως το πάγωμα από την εξάρτηση (ιβ) φέρνει την εντολή Ε5 να χρησιμοποιεί τα δεδομένα που παράγει η Ε3 με έναν κύκλο καθυστέρηση, κι επομένως η εξάρτηση (ι) δεν θα απαιτεί πια παροχέτευση για την τήρησή της. Επίσης, ο δομικός κίνδυνος στη μνήμη αναβάλλει την ανάκληση της εντολής Ε7 για δύο κύκλους, κι έτσι ούτε η εξάρτηση (ια) δεν απαιτεί πια παροχέτευση. Άρα, όπως θα φανεί στη συνέχεια και στα διαγράμματα χρονισμού, οι μόνες εξαρτήσεις που ενεργοποιούν το μηχανισμό παροχέτευσης είναι οι (η) και (ιβ).

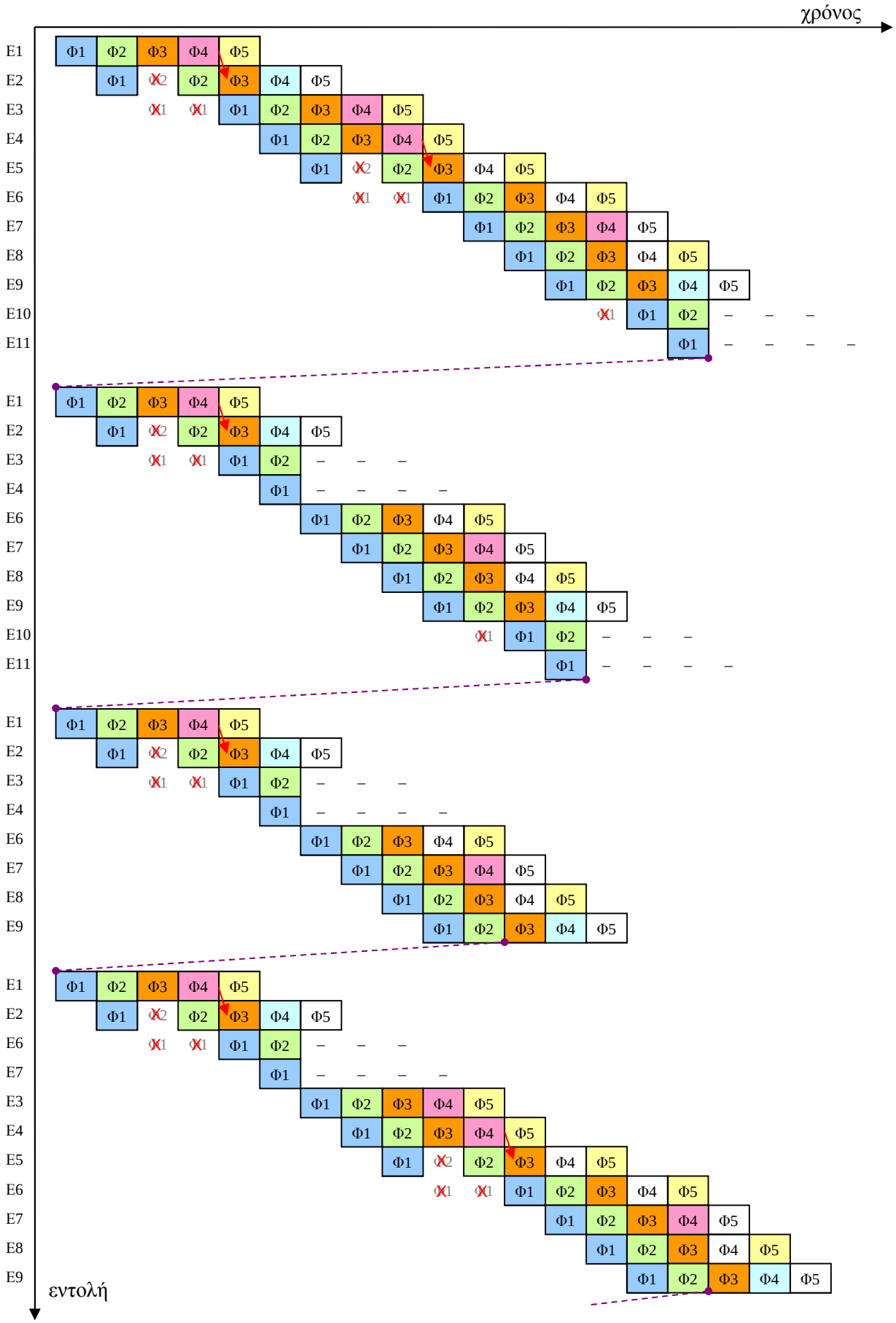
Πέρα από τις δομικές εξαρτήσεις και τις εξαρτήσεις δεδομένων, έχουμε να μελετήσουμε και τις διαδικασιακές εξαρτήσεις που δημιουργούν οι εντολές διακλάδωσης. Μια διακλάδωση δημιουργεί δύο ροές ελέγχου, μία στην κατεύθυνση εκτέλεσης του άλματος, δηλαδή προς τις εντολές που βρίσκονται στη διεύθυνση προορισμού του άλματος της διακλάδωσης, και μία στην κατεύθυνση μη εκτέλεσης άλματος, δηλαδή προς τις εντολές που βρίσκονται κάτω από την εντολή διακλάδωσης. Αν ληφθεί η μία κατεύθυνση, τότε αυτόματα δεν εκτελούνται οι εντολές της άλλης κατεύθυνσης, μέχρι το σημείο που οι δύο ροές ξανασυναντιούνται. Οι εντολές που δε γνωρίζουμε αν θα εκτελεστούν μετά από μία εντολή διακλάδωσης είναι διαδικασιακά εξαρτημένες από αυτή.
Η εύρεση των διαδικασιακά εξαρτημένων εντολών μιας διακλάδωσης δεν είναι γενικά εύκολη διαδικασία. Η πολυπλοκότητα έγκειται στο ότι διαφορετικές ροές ελέγχου πολλές φορές συναντιούνται στον κώδικα, κι έτσι δεν είναι προφανές το αν κάποιες εντολές μπορεί να μην εκτελεστούν αν ληφθεί η μία από τις δύο κατευθύνσεις μιας διακλάδωσης. Για παράδειγμα, αν η διακλάδωση έχει αρνητική μετατόπιση (δηλαδή εκτελεί άλμα προς τα πίσω στον κώδικα), δημιουργώντας μια δομή βρόχου, τότε οι εντολές στην κατεύθυνση προορισμού θα έχουν ήδη εκτελεστεί μια φορά από τη ροή ελέγχου που μπαίνει στο βρόχο, οπότε κι αν ακόμα το άλμα δεν εκτελεστεί, οι εντολές αυτές δεν θεωρούνται διαδικασιακά εξαρτημένες από την εντολή διακλάδωσης. Όμως τότε ούτε οι εντολές της άλλης κατεύθυνσης μπορούν να θεωρηθούν διαδικασιακά εξαρτημένες από τη διακλάδωση, αφού βρίσκονται στην έξοδο από το βρόχο και έτσι γνωρίζουμε ότι κάποτε θα εκτελεστούν. Για απλούστευση, σε μια τέτοια περίπτωση εξετάζουμε τις πιο κοντινές χρονικά εκτελούμενες εντολές, οπότε κι εφόσον δε γνωρίζουμε πότε θα βγούμε από το βρόχο, θα θεωρήσουμε τις εντολές που ακολουθούν τη διακλάδωση ως διαδικασιακά εξαρτημένες από αυτή.
Τα παραπάνω έχουν περισσότερο θεωρητική αξία. Στην ουσία μας ενδιαφέρουν οι κίνδυνοι που δημιουργούνται στη ΜΕΔ, και αυτοί θα εμφανιστούν όσο η εντολή διακλάδωσης δεν έχει εκτελεστεί οπότε δε μπορούμε να γνωρίζουμε με βεβαιότητα αν οι εντολές που στο μεταξύ έχουν εισέλθει στη ΜΕΔ θα εκτελεστούν ή αν θα πρέπει να ακυρωθούν. Ανάλογα με την πρόβλεψη που κάνουμε – ή δεν κάνουμε, αυτές οι εντολές μπορεί να είναι από τη μία ή από την άλλη κατεύθυνση της διακλάδωσης. Βέβαια, από αυτή την οπτική γωνία, υπάρχει το ενδεχόμενο να έχουμε διαδικασιακούς κινδύνους και προς εντολές που προηγουμένως αναφέραμε ότι δεν θεωρούμε διαδικασιακά εξαρτημένες από τη διακλάδωση! Αυτό συμβαίνει, επειδή για τους κινδύνους περιοριζόμαστε σε πολύ κοντινές χρονικά εκτελούμενες εντολές, οι οποίες αν μελετηθούν σε μεγαλύτερο εύρος εκτέλεσης μπορεί να είναι προφανές ότι σε κάποια στιγμή εκτελούνται.
Με τις εντολές διακλάδωσης να ολοκληρώνονται στη φάση Φ4 του μηχανισμού μερικής επικάλυψης, οι εντολές οι οποίες θα ξεκινούν και θα μπορεί να ακυρωθούν είναι μέχρι τρεις, με το ακριβές πλήθος να εξαρτάται από το αν μεσολαβεί κάποιος άλλος κίνδυνος, δομικός ή δεδομένων. Για παράδειγμα, με ενοποιημένη κρυφή μνήμη εντολών και δεδομένων, αν οι εντολές που προηγούνται της εντολής διακλάδωσης είναι εντολές προσπέλασης μνήμης, τότε καμία εντολή δε θα μπορέσει να ανακληθεί πριν την εκτέλεση της εντολής διακλάδωσης, κι έτσι καμία εντολή δε θα χρειαστεί να ακυρωθεί.

Στον παραπάνω κώδικα MIPS, η εντολή διακλάδωσης E2 δημιουργεί διαδικασιακή εξάρτηση προς τις εντολές E3, E4 και E5. Οι εντολές που ακολουθούν στη συνέχεια δεν είναι διαδικασιακά εξαρτημένες από την E2, αφού βρίσκονται μετά το σημείο συνάντησης των δύο ροών ελέγχου που δημιουργεί η E2 – όπου βέβαια η μία ροή είναι τετριμμένη και δεν περιέχει καμία εντολή. Και οι τρεις αυτές εντολές είναι ενδεχόμενο να εμπλακούν σε διαδικασιακό κίνδυνο αν προβλεφθεί μη εκτέλεση άλματος και τελικά το άλμα εκτελεστεί. Στην πραγματικότητα, η E5 δεν θα εμπλακεί σε τέτοιο κίνδυνο, αφού λόγω της εντολής φόρτωσης E1 και του δομικού κινδύνου μνήμης δεν θα προλάβει να ανακληθεί!
Όσο αφορά την εντολή διακλάδωσης E9, διαδικασιακά εξαρτημένες θα θεωρούσαμε τις επόμενες εντολές κάτω από τη διακλάδωση, από τις οποίες σε κίνδυνο θα μπορούσαν να εμπλακούν έως τρεις. Από τις μη εξαρτημένες εντολές της διεύθυνσης προορισμού, μόνο οι E1, E2 και E3 θα μπορούσαν να εμπλακούν σε διαδικασιακό κίνδυνο. Στην πραγματικότητα, η E3 δεν θα εμπλακεί σε τέτοιο κίνδυνο λόγω του κινδύνου δεδομένων της εξάρτησης (η), ο οποίος θα καθυστερήσει την ανάκλησή της.

Για την κατασκευή του διαγράμματος χρονισμού για τέσσερεις επαναλήψεις του βρόχου πρέπει να λάβουμε υπόψη όλα τα παραπάνω, αλλά και να μελετήσουμε την εξέλιξη στο μηχανισμό πρόβλεψης διακλαδώσεων. Σύμφωνα με την εκφώνηση, τα ψηφία ιστορίας των δύο θέσεων του πίνακα που αντιστοιχούν στις δύο διακλαδώσεις του κώδικά μας είναι αρχικά “00”. Επομένως ο μηχανισμός πρόβλεψης και για τις δύο διακλαδώσεις προβλέπει «μη εκτέλεση» στην πρώτη επανάληψη. Από τις δύο διακλαδώσεις, η πρώτη (εντολή E2) δεν εκτελεί άλμα, κι επομένως η πρόβλεψη είναι επιτυχημένη και τα ψηφία ιστορίας δεν αλλάζουν, ενώ η δεύτερη (εντολή E9) εκτελεί άλμα, κι επομένως η πρόβλεψη αποτυγχάνει, οπότε τα ψηφία ιστορίας αλλάζουν κατά τη φάση Φ4 και γίνονται “01”, ενώ ταυτόχρονα ακυρώνονται όποιες εντολές έχουν λανθασμένα ξεκινήσει. Στη δεύτερη επανάληψη οι προβλέψεις παραμένουν ίδιες, δηλαδή «μη εκτέλεση», αλλά τώρα και οι δύο διακλαδώσεις εκτελούν άλμα και οι προβλέψεις αποτυγχάνουν. Έτσι, στη Φ4 της E2 τα ψηφία ιστορίας γίνονται “01”, ενώ στη Φ4 της E9 τα ψηφία ιστορίας γίνονται “11”, ενώ έχουμε και τις αντίστοιχες ακυρώσεις εντολών. Στην τρίτη επανάληψη, η πρόβλεψη της E2 είναι ακόμα «μη εκτέλεση» και αποτυγχάνει, τα ψηφία ιστορίας γίνονται “11” και ακυρώνονται οι λανθασμένες εντολές, ενώ η πρόβλεψη της E9 είναι τώρα «εκτέλεση» και επιτυγχάνει, οπότε δεν αλλάζουν τα ψηφία ιστορίας της. Τέλος, στην τέταρτη επανάληψη, η πρόβλεψη της E2 είναι «εκτέλεση», αλλά πάλι αποτυγχάνει αφού η E2 εκτελεί άλμα, οπότε στη φάση Φ4 τα ψηφία ιστορίας γίνονται “10” και ακυρώνονται οι αντίστοιχες εντολές, ενώ η E9 εκτελείται όπως ακριβώς στην τρίτη επανάληψη.

Σύμφωνα με όλα τα παραπάνω, το ζητούμενο διάγραμμα χρονισμού δίνεται στην επόμενη σελίδα. Οι διαφορετικές φάσεις του μηχανισμού επικάλυψης δείχνονται με διαφορετικό χρώμα, ενώ με λευκό έχουμε χρωματίσει τις φάσεις Φ4 εντολών που δεν κάνουν προσπέλαση μνήμης ούτε ολοκλήρωση διακλάδωσης, καθώς και τις Φ5 εντολών που δεν κάνουν αποθήκευση αποτελέσματος στο ΦΚ. Εφόσον οι διακλαδώσεις ολοκληρώνονται στη φάση Φ4, έχουμε βάλει κατάλληλο χρώμα, ώστε να ξεχωρίσουμε τις φάσεις Φ4 που κάνουν προσπέλαση μνήμης (μωβ), από αυτές που αφορούν εντολές διακλάδωσης (γαλάζιο). Σε λάθος πρόβλεψη, ακυρώνονται μέχρι τρεις εντολές – εδώ έχουμε δύο εντολές σε όλες τις περιπτώσεις ακύρωσης – μετά τη φάση Φ4 της εντολής διακλάδωσης. Όσο αφορά τις προσπελάσεις μνήμης, έχουμε προσέξει να μην παραβιάζονται οι δομικές εξαρτήσεις μνήμης, και γι’ αυτό κάτω από φάση Φ4 εντολών μνήμης δεν υπάρχει φάση Φ1 επόμενης εντολής. Ακόμα, με βέλη δείχνουμε τα σημεία στα οποία ενεργοποιείται ο μηχανισμός παροχέτευσης. Τέλος, για εξοικονόμηση χώρου, οι διαδοχικές επαναλήψεις δείχνονται μετατοπισμένες προς τα αριστερά, με τις διακεκομμένες γραμμές να δείχνουν ακριβώς τις χρονικές στιγμές που αρχίζει κάθε επόμενη επανάληψη σε σχέση με την προηγούμενη. Ας σημειώσουμε ότι η πέμπτη επανάληψη θα αρχίσει μετά τη φάση Φ2 της εντολής E9 της τέταρτης επανάληψης λόγω του δομικού κινδύνου που δημιουργεί η εντολή αποθήκευσης E7.

Β. Οι διευθύνσεις μνήμης που η ΜΕΔ στέλνει στη ΜΔΜ παράγονται από τις φάσεις Φ1 και Φ4 του κύκλου εντολής. Ειδικότερα, η φάση Φ1, που είναι η φάση ανάκλησης, στέλνει από τον PC στη μνήμη τις διευθύνσεις των εντολών που προσκομίζονται, ενώ η φάση Φ4, που



είναι η φάση προσπέλασης μνήμης και ολοκλήρωσης διακλαδώσεων, στέλνει από τον κατα-

χωρητή επικάλυψης EX/MEM στη μνήμη τις διευθύνσεις δεδομένων που φορτώνονται ή αποθηκεύονται.
Οι διευθύνσεις ανάκλησης εντολών παράγονται σε κάθε κύκλο μηχανής, ενώ οι διευθύνσεις προσπέλασης δεδομένων παράγονται μόνο στους κύκλους που εντολές φόρτωσης και αποθήκευσης βρίσκονται στη φάση Φ4. Αν είχαμε διαχωρισμένη κρυφή μνήμη εντολών και δεδομένων, τότε είναι φανερό ότι σε κάποιους κύκλους η ΜΕΔ θα παρήγαγε δύο διευθύνσεις τόσο εντολής όσο και δεδομένων. Από αυτές η μία θα πήγαινε στη ΜΔΜ εντολών και η άλλη στη ΜΔΜ δεδομένων. Όμως με ενοποιημένη κρυφή μνήμη, όταν η ΜΕΔ παράγει δύο διευθύνσεις έχουμε δομικό κίνδυνο μνήμης, και όπως ήδη αναφέραμε, στη ΜΔΜ στέλνεται η διεύθυνση δεδομένων, ενώ η διεύθυνση εντολής παραμένει στη ΜΕΔ και στέλνεται στη ΜΔΜ μόλις πάψουν να παράγονται διευθύνσεις δεδομένων. Έτσι δημιουργούνται αντίστοιχα παγώματα στο μηχανισμό μερικής επικάλυψης.

Το διάγραμμα χρονισμού που σχηματίσαμε νωρίτερα λαμβάνει υπόψη την ενοποιημένη κρυφή μνήμη και τα παγώματα που δημιουργούνται σε περίπτωση δομικών κινδύνων μνήμης, οπότε μπορεί να μας δώσει απ' ευθείας τις ζητούμενες λογικές διευθύνσεις. Ειδικότερα, σε κάθε κύκλο μηχανής έχουμε ακριβώς μια λογική διεύθυνση, και αυτή θα είναι είτε η διεύθυνση που αντιστοιχεί στη μοναδική φάση Φ1 του κύκλου είτε, αν υπάρχει φάση Φ4 που κάνει προσπέλαση μνήμης, η διεύθυνση που αντιστοιχεί στη φάση αυτή.
Οι διευθύνσεις που προκύπτουν από τις φάσεις Φ1 είναι οι διευθύνσεις που έχουν οι εντολές στη μνήμη. Εφόσον μας δίνεται ότι αρχική τιμή του PC είναι η τιμή 0x1001a0c0, αυτό σημαίνει ότι η διεύθυνση της πρώτης εντολής που ανακαλούμε, δηλαδή της E1, είναι η 0x1001a0c0. Έτσι, με δεδομένη την αύξηση κατά 4 των διευθύνσεων των επόμενων εντολών, αυτές θα είναι: 0x1001a0c4 της E2, 0x1001a0c8 της E3, 0x1001a0cc της E4, 0x1001a0d0 της E5, 0x1001a0d4 της E6, 0x1001a0d8 της E7, 0x1001a0dc της E8, 0x1001a0e0 της E9, 0x1001a0e4 της E10 και 0x1001a0e8 της E11, έχοντας συμπεριλάβει και τις εντολές E10 και E11 που δεν δίνονται, αλλά συμμετέχουν στη συγκεκριμένη διαδικασία. Υποθέτοντας μια παραδοσιακή στατική κρυφή μνήμη, οι λογικές διευθύνσεις των εντολών είναι πάντα οι ίδιες για όλη την εκτέλεση του κώδικα.
Οι διευθύνσεις που προκύπτουν από τις φάσεις Φ4 πρέπει να υπολογιστούν από το άθροισμα της τιμής του καταχωρητή βάσης με την αντίστοιχη μετατόπιση. Τώρα, αν και η τιμή της μετατόπισης δεν μπορεί να αλλάξει για την ίδια εντολή, σε κάθε εκτέλεση της εντολής είναι δυνατό να έχουμε διαφορετική τιμή στον καταχωρητή βάσης. Γι' αυτό και θα πρέπει να μελετήσουμε προσεκτικά την εκτέλεση του κώδικα, ώστε να εντοπίσουμε σημεία στα οποία αλλάζει αυτή η τιμή. Στον κώδικά μας λοιπόν, οι εντολές προσπέλασης μνήμης είναι οι E1, E3, E4 και E7. Από αυτές, οι τρεις πρώτες έχουν τον ίδιο καταχωρητή βάσης $4, ενώ η τέταρτη έχει καταχωρητή βάσης τον $5. Παρατηρούμε ότι η εντολή E6 αυξάνει την τιμή του $4 κατά 12, ενώ η εντολή E8 αυξάνει την τιμή του $5 κατά 4. Καμία άλλη εντολή δεν μεταβάλλει τις τιμές των δύο αυτών καταχωρητών. Έτσι, με αρχική τιμή 0x1006a0b0, ο $4 μετά την εντολή E6 της πρώτης επανάληψης θα έχει τιμή 0x1006a0bc, μετά την ίδια εντολή της δεύτερης επανάληψης θα έχει τιμή 0x1006a0c8, μετά την ίδια εντολή της τρίτης επανάληψης θα έχει τιμή 0x1006a0d4, ενώ μετά την ίδια εντολή της τέταρτης επανάληψης θα έχει τιμή 0x1006a0e0. Από την άλλη μεριά, με αρχική τιμή 0x1008a0c0, ο $5μετά την εντολή E8 της πρώτης επανάληψης θα έχει τιμή 0x1008a0c4, μετά την ίδια εντολή της δεύτερης επανάληψης θα έχει τιμή 0x1008a0c8, μετά την ίδια εντολή της τρίτης επανάληψης θα έχει τιμή 0x1008a0cc, ενώ μετά την ίδια εντολή της τέταρτης επανάληψης θα έχει τιμή 0x1008a0d0.
Στην πρώτη λοιπόν επανάληψη εκτελούνται και οι τέσσερεις προσπελάσεις μνήμης δεδομένων. Η προσπέλαση που γίνεται στη φάση Φ4 της E1 θα γίνεται στη διεύθυνση 0+$4, δηλαδή στη διεύθυνση 0x1006a0b0. Η προσπέλαση που γίνεται στη φάση Φ4 της E3 θα γίνεται στη διεύθυνση 4+$4, δηλαδή στη διεύθυνση 0x1006a0b4. Η προσπέλαση που γίνεται στη φάση Φ4 της E4 θα γίνεται στη διεύθυνση 8+$4, δηλαδή στη διεύθυνση 0x1006a0b8. Τέλος, η προσπέλαση που γίνεται στη φάση Φ4 της E7 θα γίνεται στη διεύθυνση 0+$5, δηλαδή στη διεύθυνση 0x1008a0c0. Στη δεύτερη επανάληψη η E2 εκτελεί άλμα, κι επομένως δεν εκτελούνται οι προσπελάσεις δεδομένων των εντολών E3 και E4. Και οι δύο αυτές εντολές έχουν προλάβει να ανακληθούν, αλλά δεν έχουν προλάβει να φτάσουν στη φάση Φ4 πριν ακυρω-

θούν. Έτσι, οι μόνες εντολές που κάνουν προσπέλαση μνήμης δεδομένων είναι οι Ε1 και Ε7 στις αντίστοιχες φάσεις Φ4, με αντίστοιχες διευθύνσεις 0x1006a0bc και 0x1008a0c4. Η ίδια εξέλιξη υπάρχει και στην τρίτη επανάληψη, όπου οι διευθύνσεις προσπέλασης μνήμης των Φ4 των εντολών Ε1 και Ε7 θα είναι 0x1006a0c8 και 0x1008a0c8 αντίστοιχα. Στην τέταρτη επανάληψη εκτελούνται όλες οι προσπελάσεις μνήμης δεδομένων. Εδώ έχουμε πρόβλεψη εκτέλεσης άλματος στην Ε2, με αποτέλεσμα την λανθασμένη ανάκληση της Ε7, όμως η εντολή ακυρώνεται πριν τη φάση Φ4. Έτσι τελικά εκτελούνται οι φάσεις Φ4 των εντολών Ε1, Ε4, Ε4 και Ε7 – η οποία ξαναεμφανίζεται στη σωστή πια εκτέλεσή της – με αντίστοιχες διευθύνσεις προσπέλασης 0x1006a0d4, 0x1006a0d8, 0x1006a0dc και 0x1008a0cc.

Συνολικά, η ακολουθία διευθύνσεων που στέλνονται στη ΜΔΜ της ενοποιημένης κρυφής μνήμης, με βάση τους κύκλους μηχανής του διαγράμματος χρονισμού, θα είναι:

Φ1(Ε1): 0x1001a0c0 – 1ος κύκλος
Φ1(Ε2): 0x1001a0c4 – 2ος κύκλος
Φ1(Ε3): 0x1001a0c8 – 3ος κύκλος, δεν ολοκληρώνεται λόγω παγώματος
Φ4(Ε1): 0x1006a0b0 – 4ος κύκλος
Φ1(Ε3): 0x1001a0c8 – 5ος κύκλος
Φ1(Ε4): 0x1001a0cc – 6ος κύκλος
Φ1(Ε5): 0x1001a0d0 – 7ος κύκλος
Φ4(Ε3): 0x1006a0b4 – 8ος κύκλος
Φ4(Ε4): 0x1006a0b8 – 9ος κύκλος
Φ1(Ε6): 0x1001a0d4 – 10ος κύκλος
Φ1(Ε7): 0x1001a0d8 – 11ος κύκλος
Φ1(Ε8): 0x1001a0dc – 12ος κύκλος
Φ1(Ε9): 0x1001a0e0 – 13ος κύκλος
Φ4(Ε7): 0x1008a0c0 – 14ος κύκλος
Φ1(Ε10): 0x1001a0e4 – 15ος κύκλος, αν και η εντολή αργότερα ακυρώνεται
Φ1(Ε11): 0x1001a0e8 – 16ος κύκλος, αν και η εντολή ακυρώνεται
Φ1(Ε1): 0x1001a0c0 – 17ος κύκλος
Φ1(Ε2): 0x1001a0c4 – 18ος κύκλος
Φ1(Ε3): 0x1001a0c8 – 19ος κύκλος, δεν ολοκληρώνεται λόγω παγώματος
Φ4(Ε1): 0x1006a0bc – 20ος κύκλος
Φ1(Ε3): 0x1001a0c8 – 21ος κύκλος, αν και η εντολή αργότερα ακυρώνεται
Φ1(Ε4): 0x1001a0cc – 22ος κύκλος, αν και η εντολή ακυρώνεται
Φ1(Ε6): 0x1001a0d4 – 23ος κύκλος
Φ1(Ε7): 0x1001a0d8 – 24ος κύκλος
Φ1(Ε8): 0x1001a0dc – 25ος κύκλος
Φ1(Ε9): 0x1001a0e0 – 26ος κύκλος
Φ4(Ε7): 0x1008a0c4 – 27ος κύκλος
Φ1(Ε10): 0x1001a0e4 – 28ος κύκλος, αν και η εντολή αργότερα ακυρώνεται
Φ1(Ε11): 0x1001a0e8 – 29ος κύκλος, αν και η εντολή ακυρώνεται
Φ1(Ε1): 0x1001a0c0 – 30ος κύκλος
Φ1(Ε2): 0x1001a0c4 – 31ος κύκλος
Φ1(Ε3): 0x1001a0c8 – 32ος κύκλος, δεν ολοκληρώνεται λόγω παγώματος
Φ4(Ε1): 0x1006a0c8 – 33ος κύκλος
Φ1(Ε3): 0x1001a0c8 – 34ος κύκλος, αν και η εντολή αργότερα ακυρώνεται
Φ1(Ε4): 0x1001a0cc – 35ος κύκλος, αν και η εντολή ακυρώνεται
Φ1(Ε6): 0x1001a0d4 – 36ος κύκλος
Φ1(Ε7): 0x1001a0d8 – 37ος κύκλος
Φ1(Ε8): 0x1001a0dc – 38ος κύκλος
Φ1(Ε9): 0x1001a0e0 – 39ος κύκλος
Φ4(Ε7): 0x1008a0c8 – 40ος κύκλος
Φ1(Ε1): 0x1001a0c0 – 41ος κύκλος
Φ1(Ε2): 0x1001a0c4 – 42ος κύκλος
Φ1(Ε6): 0x1001a0d4 – 43ος κύκλος, δεν ολοκληρώνεται λόγω παγώματος

Φ4(Ε1): 0x1006a0d4 – 44ος κύκλος
Φ1(Ε6): 0x1001a0d4 – 45ος κύκλος, αν και η εντολή αργότερα ακυρώνεται
Φ1(Ε7): 0x1001a0d8 – 46ος κύκλος, αν και η εντολή ακυρώνεται
Φ1(Ε3): 0x1001a0c8 – 47ος κύκλος
Φ1(Ε4): 0x1001a0cc – 48ος κύκλος
Φ1(Ε5): 0x1001a0d0 – 49ος κύκλος
Φ4(Ε3): 0x1006a0d8 – 50ος κύκλος
Φ4(Ε4): 0x1006a0dc – 51ος κύκλος
Φ1(Ε6): 0x1001a0d4 – 52ος κύκλος
Φ1(Ε7): 0x1001a0d8 – 53ος κύκλος
Φ1(Ε8): 0x1001a0dc – 54ος κύκλος
Φ1(Ε9): 0x1001a0e0 – 55ος κύκλος
Φ4(Ε7): 0x1008a0cc – 56ος κύκλος

Έχοντας βρει την ακολουθία διευθύνσεων που στέλνονται στην κρυφή μνήμη, είναι πια εύκολη η ανάλυση των προσπελάσεων. Για μέγεθος 128KB και πλαίσια των 16 bytes, η κρυφή μνήμη θα περιέχει 128K/16 = 8K πλαίσια, που για συνολοσυσχετιστική οργάνωση δύο δρόμων θα αντιστοιχούν σε 4K = 2^{12} σύνολα. Έτσι από τα 32 bits ή τα 8 δεκαεξαδικά ψηφία των διευθύνσεων, τα 4 λιγότερο σημαντικά bits ή το 1 λιγότερο σημαντικό δεκαεξαδικό ψηφίο θα είναι η μετατόπιση στο πλαίσιο, τα επόμενα 12 bits ή 3 δεκαεξαδικά ψηφία θα είναι ο αριθμοδείκτης συνόλου, και τα υπόλοιπα 16 bits ή 4 δεκαεξαδικά ψηφία θα είναι το πεδίο ετικέτας. Διατηρώντας τη δεκαεξαδική αναπαράσταση για λόγους ευκολίας, οι προσπελάσεις μνήμης αναλύονται με βάση τον ακόλουθο πίνακα:

cc	Διεύθυνση	Σύνολο	Ετικέτα	Περιγραφή προσπέλασης	LRU
1	0x1001a0c0	0xa0c	0x1001	Αστοχία, τοποθέτηση στο 1ο πλαίσιο	-
2	0x1001a0c4	0xa0c	0x1001	Ευστοχία στο 1ο πλαίσιο	-
3	0x1001a0c8	0xa0c	0x1001	Ευστοχία στο 1ο πλαίσιο	-
4	0x1006a0b0	0xa0b	0x1006	Αστοχία, τοποθέτηση στο 1ο πλαίσιο	-
5	0x1001a0c8	0xa0c	0x1001	Ευστοχία στο 1ο πλαίσιο	-
6	0x1001a0cc	0xa0c	0x1001	Ευστοχία στο 1ο πλαίσιο	-
7	0x1001a0d0	0xa0d	0x1001	Αστοχία, τοποθέτηση στο 1ο πλαίσιο	-
8	0x1006a0b4	0xa0b	0x1006	Ευστοχία στο 1ο πλαίσιο	-
9	0x1006a0b8	0xa0b	0x1006	Ευστοχία στο 1ο πλαίσιο	-
10	0x1001a0d4	0xa0d	0x1001	Ευστοχία στο 1ο πλαίσιο	-
11	0x1001a0d8	0xa0d	0x1001	Ευστοχία στο 1ο πλαίσιο	-
12	0x1001a0dc	0xa0d	0x1001	Ευστοχία στο 1ο πλαίσιο	-
13	0x1001a0e0	0xa0e	0x1001	Αστοχία, τοποθέτηση στο 1ο πλαίσιο	-
14	0x1008a0c0	0xa0c	0x1008	Αστοχία, τοποθέτηση στο 2ο πλαίσιο	1ο
15	0x1001a0e4	0xa0e	0x1001	Ευστοχία στο 1ο πλαίσιο	-
16	0x1001a0e8	0xa0e	0x1001	Ευστοχία στο 1ο πλαίσιο	-
17	0x1001a0c0	0xa0c	0x1001	Ευστοχία στο 1ο πλαίσιο	2ο
18	0x1001a0c4	0xa0c	0x1001	Ευστοχία στο 1ο πλαίσιο	2ο
19	0x1001a0c8	0xa0c	0x1001	Ευστοχία στο 1ο πλαίσιο	2ο
20	0x1006a0bc	0xa0b	0x1006	Ευστοχία στο 1ο πλαίσιο	-
21	0x1001a0c8	0xa0c	0x1001	Ευστοχία στο 1ο πλαίσιο	2ο
22	0x1001a0cc	0xa0c	0x1001	Ευστοχία στο 1ο πλαίσιο	2ο
23	0x1001a0d4	0xa0d	0x1001	Ευστοχία στο 1ο πλαίσιο	-
24	0x1001a0d8	0xa0d	0x1001	Ευστοχία στο 1ο πλαίσιο	-
25	0x1001a0dc	0xa0d	0x1001	Ευστοχία στο 1ο πλαίσιο	-
26	0x1001a0e0	0xa0e	0x1001	Ευστοχία στο 1ο πλαίσιο	-
27	0x1008a0c4	0xa0c	0x1008	Ευστοχία στο 2ο πλαίσιο	1ο
28	0x1001a0e4	0xa0e	0x1001	Ευστοχία στο 1ο πλαίσιο	-
29	0x1001a0e8	0xa0e	0x1001	Ευστοχία στο 1ο πλαίσιο	-

30	0x1001a0c0	0xa0c	0x1001	Ευστοχία στο 1^{ο} πλαίσιο	2^{ο}
31	0x1001a0c4	0xa0c	0x1001	Ευστοχία στο 1^{ο} πλαίσιο	2^{ο}
32	0x1001a0c8	0xa0c	0x1001	Ευστοχία στο 1^{ο} πλαίσιο	2^{ο}
33	0x1006a0c8	0xa0c	0x1006	Αστοχία, αντικατάσταση 2^{ου} πλαισίου	1^{ο}
34	0x1001a0c8	0xa0c	0x1001	Ευστοχία στο 1^{ο} πλαίσιο	2^{ο}
35	0x1001a0cc	0xa0c	0x1001	Ευστοχία στο 1^{ο} πλαίσιο	2^{ο}
36	0x1001a0d4	0xa0d	0x1001	Ευστοχία στο 1^{ο} πλαίσιο	-
37	0x1001a0d8	0xa0d	0x1001	Ευστοχία στο 1^{ο} πλαίσιο	-
38	0x1001a0dc	0xa0d	0x1001	Ευστοχία στο 1^{ο} πλαίσιο	-
39	0x1001a0e0	0xa0e	0x1001	Ευστοχία στο 1^{ο} πλαίσιο	-
40	0x1008a0c8	0xa0c	0x1008	Αστοχία, αντικατάσταση 2^{ου} πλαισίου	1^{ο}
41	0x1001a0c0	0xa0c	0x1001	Ευστοχία στο 1^{ο} πλαίσιο	2^{ο}
42	0x1001a0c4	0xa0c	0x1001	Ευστοχία στο 1^{ο} πλαίσιο	2^{ο}
43	0x1001a0d4	0xa0d	0x1001	Ευστοχία στο 1^{ο} πλαίσιο	-
44	0x1006a0d4	0xa0d	0x1006	Αστοχία, τοποθέτηση στο 2^{ο} πλαίσιο	1^{ο}
45	0x1001a0d4	0xa0d	0x1001	Ευστοχία στο 1^{ο} πλαίσιο	2^{ο}
46	0x1001a0d8	0xa0d	0x1001	Ευστοχία στο 1^{ο} πλαίσιο	2^{ο}
47	0x1001a0c8	0xa0c	0x1001	Ευστοχία στο 1^{ο} πλαίσιο	2^{ο}
48	0x1001a0cc	0xa0c	0x1001	Ευστοχία στο 1^{ο} πλαίσιο	2^{ο}
49	0x1001a0d0	0xa0d	0x1001	Ευστοχία στο 1^{ο} πλαίσιο	2^{ο}
50	0x1006a0d8	0xa0d	0x1006	Ευστοχία στο 2^{ο} πλαίσιο	1^{ο}
51	0x1006a0dc	0xa0d	0x1006	Ευστοχία στο 2^{ο} πλαίσιο	1^{ο}
52	0x1001a0d4	0xa0d	0x1001	Ευστοχία στο 1^{ο} πλαίσιο	2^{ο}
53	0x1001a0d8	0xa0d	0x1001	Ευστοχία στο 1^{ο} πλαίσιο	2^{ο}
54	0x1001a0dc	0xa0d	0x1001	Ευστοχία στο 1^{ο} πλαίσιο	2^{ο}
55	0x1001a0e0	0xa0e	0x1001	Ευστοχία στο 1^{ο} πλαίσιο	-
56	0x1008a0cc	0xa0c	0x1008	Ευστοχία στο 2^{ο} πλαίσιο	1^{ο}

Η τελευταία στήλη δείχνει ποιο πλαίσιο από τα δύο του συνόλου είναι το *νέο* LRU, το οποίο και θα αντικατασταθεί στην *επόμενη* αστοχία του συνόλου. Αν δεν χρησιμοποιούνται και τα δύο πλαίσια του συνόλου, η ένδειξη LRU δεν έχει νόημα.
Ας σημειωθεί ότι επειδή έχουμε ετερόχρονη εγγραφή με προσκόμιση κατά την εγγραφή, οι προσπελάσεις αποθήκευσης της εντολής Ε7 έχουν ίδια αντιμετώπιση με τις προσπελάσεις φόρτωσης των εντολών Ε1, Ε3 και Ε4. Αν δεν είχαμε προσκόμιση κατά την εγγραφή, οι προσπελάσεις της εντολής Ε7 σε περίπτωση αστοχίας δεν θα έφερναν το μπλοκ από το επόμενο επίπεδο της ιεραρχίας της μνήμης.

Γ. Για να υπολογίσουμε το μέσο χρόνο ολοκλήρωσης διαδοχικών επαναλήψεων του βρόχου, πρέπει να απαριθμήσουμε όλες τις περιπτώσεις διαφορετικού χρονισμού κατά την εκτέλεση μιας επανάληψης και να βρούμε τις πιθανότητες εμφάνισής τους. Ο χρονισμός των εντολών μιας επανάληψης αλλάζει όταν αλλάζει τόσο η πρόβλεψη μιας εντολής διακλάδωσης, όσο και το αποτέλεσμα της εκτέλεσής της. Για μεγάλο αριθμό επαναλήψεων, η διακλάδωση Ε9 έχει αμελητέα επίδραση στο μέσο χρόνο ολοκλήρωσης μιας επανάληψης, αφού σε όλες εκτός από ελάχιστες επαναλήψεις θα έχει επιτυχημένη πρόβλεψη εκτέλεσης άλματος. Θα αρκεστούμε επομένως στην ανάλυση των περιπτώσεων που αφορούν τη διακλάδωση Ε2:

1. Πρόβλεψη μη εκτέλεσης άλματος με επιτυχία
2. Πρόβλεψη μη εκτέλεσης άλματος με αποτυχία
3. Πρόβλεψη εκτέλεσης άλματος με επιτυχία
4. Πρόβλεψη εκτέλεσης άλματος με αποτυχία

Έστω ότι οι παραπάνω περιπτώσεις εμφανίζονται με πιθανότητες p1, p2, p3 και p4 αντίστοιχα. Κάθε περίπτωση συνδυάζει μια πιθανότητα πρόβλεψης με μια πιθανότητα επιτυχίας. Έστω $p_{με}$ η πιθανότητα επιτυχίας πρόβλεψης μη εκτέλεσης και $p_{ε}$ η πιθανότητα επιτυχίας πρόβλεψης εκτέλεσης άλματος. Εφόσον η επιτυχία σε όλες τις προβλέψεις είναι ομοιόμορφη, οι δύο πιθανότητες θα είναι ίσες, έστω $p_{με} = p_{ε} = p$. Τότε:

p1 = p(πρόβλεψη μη εκτέλεσης άλματος)*$p_{με}$ = 0,7*p
p2 = p(πρόβλεψη μη εκτέλεσης άλματος)*(1-$p_{με}$) = 0,7*(1-p)
p3 = p(πρόβλεψη εκτέλεσης άλματος)*$p_{ε}$ = 0,3*p
p4 = p(πρόβλεψη εκτέλεσης άλματος)*(1-$p_{ε}$) = 0,3*(1-p)

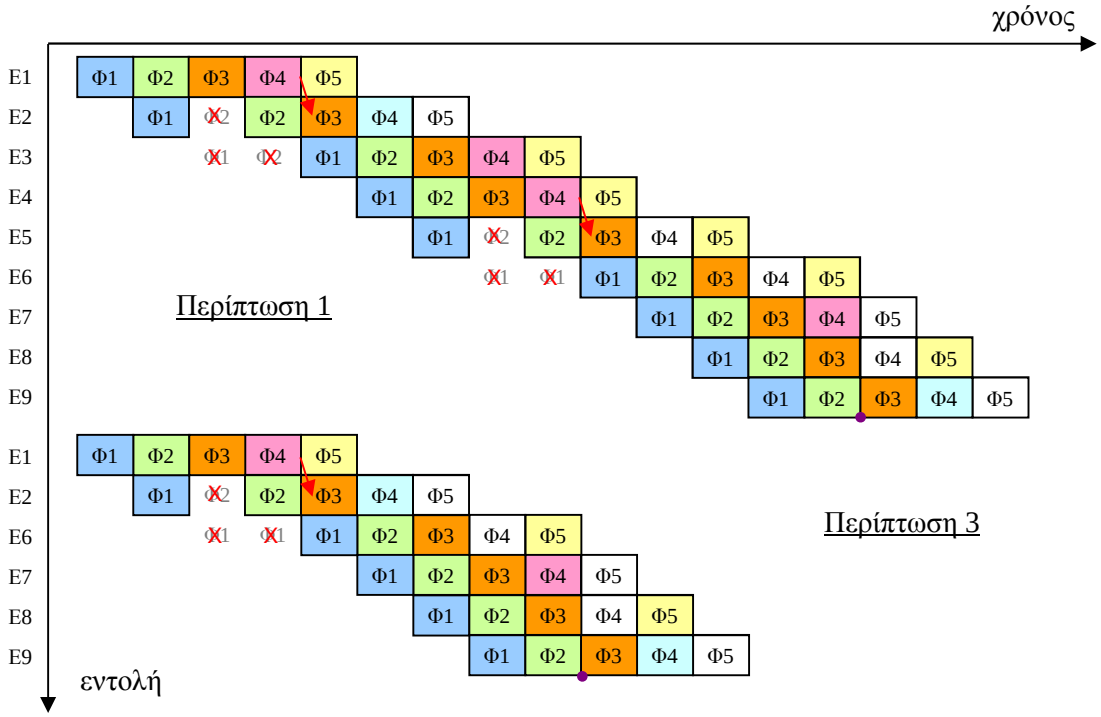
Για να βρούμε την πιθανότητα p από το ποσοστό εκτέλεσης άλματος που μας δίνεται, παρατηρούμε ότι εκτέλεση άλματος μπορούμε να έχουμε είτε όταν έχουμε επιτυχημένη πρόβλεψη εκτέλεσης άλματος είτε όταν έχουμε αποτυχημένη πρόβλεψη μη εκτέλεσης άλματος. Οπότε:

40% = p3 + p2 = 0,3*p + 0,7*(1-p) ⇒ 0,4*p = 0,3 ⇒ p = 75%

Άρα:

p1 = 52,5%
p2 = 17,5%
p3 = 22,5%
p4 = 7,5%

Για να υπολογίσουμε τους χρόνους σε αριθμό κύκλων μηχανής που αντιστοιχούν στις τέσσερεις περιπτώσεις, θα χρησιμοποιήσουμε διαγράμματα χρονισμού, από τα οποία θα μετρήσουμε τους ζητούμενους αριθμούς κύκλων. Παρατηρούμε κατ' αρχήν ότι οι δύο περιπτώσεις καλύπτονται από το διάγραμμα χρονισμού που κατασκευάσαμε νωρίτερα. Ειδικότερα, η περίπτωση 2 καλύπτεται από την τρίτη επανάληψη, ενώ η περίπτωση 4 καλύπτεται από την τέταρτη επανάληψη. Επιπλέον, η πρώτη επανάληψη αντιστοιχεί μεν στην περίπτωση 1, όμως με αποτυχημένη πρόβλεψη μη εκτέλεσης άλματος της Ε9. Στην ουσία το διάγραμμα που θέλουμε για την περίπτωση 1 θα είναι ταυτόσημο, χωρίς τις τελευταίες δύο εντολές που ανακλήθηκαν λανθασμένα και διαγράφτηκαν. Για να βρούμε όμως με βεβαιότητα το χρόνο που απαιτεί η συγκεκριμένη περίπτωση, ξανακάνουμε εκ νέου το διάγραμμα χρονισμού. Κατασκευάζουμε επίσης και το διάγραμμα για την περίπτωση 3 που ούτως ή άλλως δεν διαθέτουμε. Τα δύο συμπληρωματικά διαγράμματα δίνονται στη συνέχεια:



Με βάση τα διαγράμματα χρονισμού, οι χρόνοι T1, T2, T3 και T4 των αντίστοιχων τεσσάρων περιπτώσεων θα είναι:

T1 = 14cc
T2 = 11cc
T3 = 9cc
T4 = 16cc

όπου μετράμε από την φάση Φ1 της Ε1 μέχρι και την τελευταία φάση πριν τη Φ1 της Ε1 της επόμενης επανάληψης. Το σημείο όπου αρχίζει η επόμενη επανάληψη το υποδεικνύουμε με τη μωβ τελεία.

Οι παραπάνω χρόνοι όμως δε λαμβάνουν υπόψη τις αστοχίες στην κρυφή μνήμη. Για ομοιόμορφη αστοχία, είτε σε προσπέλαση εντολής είτε σε προσπέλαση δεδομένων, το ποσοστό ευστοχίας που μας δίνεται θα αναφέρεται σε κάθε κύκλο μηχανής, αφού προσπέλαση μνήμης – όπως είδαμε και νωρίτερα – έχουμε σε κάθε κύκλο μηχανής. Έτσι, σε κάθε κύκλο θα προσθέτουμε επιβάρυνση (1-0,98)*6 = 0,12cc, οπότε οι τελικοί χρόνοι για κάθε περίπτωση θα είναι:

T1 = 14*1,12 = 15,68cc
T2 = 11*1,12 = 12,32cc
T3 = 9*1,12 = 10,08cc
T4 = 16*1,12 = 17,92cc

Επομένως ο μέσος χρόνος ολοκλήρωσης διαδοχικών επαναλήψεων θα είναι:

$T_{μέσος}$ = p1*T1 + p2*T2 + p3*T3 + p4*T4 =
= 0,525*15,68 + 0,175*12,32 + 0,225*10,08 + 0,075*17,92 =
= 14cc