Πανεπιστήμιο Θεσσαλίας
Τμήμα Πληροφορικής

Οργάνωση Η/Υ

Ενότητα 6η: Μερική Επικάλυψη Εντολών

Άσκηση 1:

Έστω μια αρχιτεκτονική μερικά επικαλυπτόμενων εντολών, όμοια με αυτή που μελετήσαμε στο μάθημα, η οποία εκτελεί ένα πρόγραμμα εφαρμογής.
Το πρόγραμμα αυτό αποτελείται από το ακόλουθο μείγμα διαφορετικών τύπων εντολών μηχανής:

εντολές ΑΛΜ 50%
εντολές μνήμης 30%
εντολές άλματος 15%
άλλες εντολές 5%

Οι εντολές της τελευταίας κατηγορίας δεν επηρεάζουν την απόδοση επικάλυψης.
Αγνοώντας εξαρτήσεις δεδομένων, βρείτε την επιτάχυνση επικάλυψης (α) με κοινή και (β) με διαχωρισμένη κρυφή μνήμη εντολών και δεδομένων, σε σχέση με μια αρχιτεκτονική χωρίς επικάλυψη, όπου όλες οι εντολές εκτελούνται στον ίδιο χρόνο.
Υποθέστε ότι οι εντολές άλματος επηρεάζουν ομοιόμορφα την τεχνική επικάλυψης, με πάγωμα της ΜΕΔ μέχρι την εκτέλεσή τους.
Θεωρήστε ιδανικό διαχωρισμό φάσεων.

Απάντηση:

Η αρχιτεκτονική που μελετήσαμε στο μάθημα εκτελεί κάθε εντολή μηχανής σε πέντε φάσεις με την τεχνική μερικής επικάλυψης διαδοχικών εντολών.
Αν κάθε φάση ολοκληρώνεται σε χρόνο τ, μια εντολή εκτελείται σε χρόνο ίσο με 5τ. Υποθέτοντας ιδανικό διαχωρισμό φάσεων, ο χρόνος εκτέλεσης T_0 μιας εντολής σε αντίστοιχη αρχιτεκτονική χωρίς επικάλυψη διαδοχικών εντολών θα είναι ίσος με τον παραπάνω, δηλαδή:

$$T_0 = 5\tau$$

Έστω ότι το πρόγραμμα εφαρμογής εκτελεί συνολικά Ν εντολές. Τότε ο συνολικός χρόνος εκτέλεσης χωρίς επικάλυψη $T_{σειρ}$ θα είναι:

$$T_{σειρ} = N \times T_0 = 5\tau N$$

ενώ ο συνολικός χρόνος εκτέλεσης Ν εντολών με ιδανική επικάλυψη θα είναι:

$$T_{επικ}^{(ιδαν)} = T_0 + \tau \times (N-1) + = 4\tau + \tau N$$

Λόγω των εξαρτήσεων μεταξύ διαδοχικών εντολών, η επικάλυψη μπορεί να μην είναι ιδανική. Πιο συγκεκριμένα, η απόδοση επικάλυψης βλάπτεται σε τρεις περιπτώσεις:

1. Στην περίπτωση των δομικών κινδύνων, στους οποίους η ανάγκη κοινής χρήσης μιας υπομονάδας της ΜΕΔ από διαδοχικές εντολές αναγκάζει τις νεώτερες εντολές να περιμένουν την απελευθέρωση της υπομονάδας από την αρχαιότερη εντολή.
2. Στην περίπτωση των κινδύνων λόγω εξαρτήσεων από δεδομένα, στους οποίους ένα ή περισσότερα τελούμενα μιας εντολής εξαρτώνται από προηγούμενες εντολές. Στη χειρότερη περίπτωση, η εντολή αυτή περιμένει μέχρι την ολοκλήρωση των τελευταίων.
3. Στην περίπτωση των κινδύνων λόγω διαδικασιακών εξαρτήσεων, στους οποίους η ροή εκτέλεσης του κώδικα αλλάζει με κάποιο άλμα. Οι εντολές που ακολουθούν το άλμα είναι εξαρτημένες διαδικασιακά από την εντολή άλματος. Στη χειρότερη περίπτωση, θα

πρέπει αυτές να περιμένουν την εκτέλεση της τελευταίας, πριν μπορέσουν να εισέλθουν στην ΜΕΔ[1].

Στην αρχιτεκτονική που χρησιμοποιούμε υποθέτουμε ότι οι μόνοι δομικοί κίνδυνοι που εμφανίζονται σχετίζονται με τη χρήση κοινής μνήμης εντολών και δεδομένων. Οι πληροφορίες που μας δίνονται δε μας επιτρέπουν να κάνουμε περισσότερες υποθέσεις για άλλους δομικούς κινδύνους. Επίσης, για τους σκοπούς αυτής της άσκησης, αγνοούμε πλήρως τις εξαρτήσεις από δεδομένα. Τέλος, όσο αφορά κινδύνους λόγω διαδικασιακών εξαρτήσεων, θεωρούμε ότι η ομοιόμορφη επίδραση αυτών στην απόδοση της τεχνικής μερικής επικάλυψης οφείλεται στο πάγωμα της διαδικασίας επικάλυψης για όλα τα άλματα και μέχρι αυτά να ολοκληρώσουν τη φάση εκτέλεσής τους.

Σύμφωνα με τα παραπάνω, και υποθέτοντας ότι όλα τα άλματα, είτε με είτε χωρίς συνθήκη, εκτελούνται στη φάση εκτέλεσης, κάθε εντολή άλματος θα εισάγει δύο παγώματα στην εκτέλεση των επόμενων εντολών. Ειδικότερα, η αμέσως επόμενη εντολή αρχίζει τη φάση ανάκλησης, μόνο μετά την ολοκλήρωση της φάσης εκτέλεσης της προηγούμενης εντολής άλματος. Έτσι, ο συνολικός χρόνος εκτέλεσης με μερική επικάλυψη με συνυπολογισμό των διαδικασιακών εξαρτήσεων – αλλά όχι ακόμα των δομικών – γίνεται:

$$T_{επικ}^{(διαδ)} = T_{επικ}^{(ιδαν)} + 15\% \times N \times 2\tau \Rightarrow T_{επικ}^{(διαδ)} = 4\tau + \tau N + 0{,}3\tau N = 4\tau + 1{,}3\tau N$$

Ο παραπάνω χρόνος $T_{επικ}^{(διαδ)}$ είναι ο χρόνος εκτέλεσης του προγράμματος εφαρμογής, εάν δεν έχουμε δομικές εξαρτήσεις. Έτσι, για την περίπτωση που έχουμε διαχωρισμένη κρυφή μνήμη εντολών και δεδομένων, η επικάλυψη περιορίζεται μόνο από τις διαδικασιακές εξαρτήσεις, κι επομένως ο χρόνος εκτέλεσης του προγράμματος θα είναι ίσος με $T_{επικ}^{(διαδ)}$.

Στην περίπτωση που έχουμε κοινή κρυφή μνήμη εντολών και δεδομένων, κάθε εντολή μνήμης εισάγει ένα ακόμα πάγωμα στη φάση ανάκλησης μιας επόμενης εντολής. Πιο συγκεκριμένα, αν η εντολή μνήμης είναι η εντολή i, τότε η εντολή που θα αναγκαστεί να περιμένει για έναν κύκλο μηχανής είναι η i+3, δεδομένου ότι η εντολή i+1 βρίσκεται ήδη στη φάση εκτέλεσης και η εντολή i+2 βρίσκεται ήδη στη φάση αποκωδικοποίησης. Για απλούστευση της ανάλυσης, υποθέτουμε ότι καμία από τις δύο εντολές που ακολουθούν μια εντολή μνήμης δεν είναι εντολή άλματος[2]. Έτσι ο συνολικός χρόνος εκτέλεσης του προγράμματος θα είναι:

$$T_{επικ} = T_{επικ}^{(διαδ)} + 30\% \times N \times \tau \Rightarrow T_{επικ} = 4\tau + 1{,}3\tau N + 0{,}3\tau N = 4\tau + 1{,}6\tau N$$

Οι αντίστοιχες επιταχύνσεις επικάλυψης για τους παραπάνω χρόνους εκτέλεσης του προγράμματος των N εντολών θα είναι:

$$Sp^{(δ)} = T_{σειρ} / T_{επικ}^{(διαδ)} \Rightarrow Sp^{(δ)} = 5\tau N / (4\tau + 1{,}3\tau N) \cong 5\tau N / 1{,}3\tau N = 3{,}846$$
$$Sp^{(κ)} = T_{σειρ} / T_{επικ} \Rightarrow Sp^{(κ)} = 5\tau N / (4\tau + 1{,}6\tau N) \cong 5\tau N / 1{,}6\tau N = 3{,}125$$

όπου η πρώτη αναφέρεται στην περίπτωση διαχωρισμένης και η δεύτερη στην περίπτωση κοινής κρυφής μνήμης εντολών και δεδομένων.

Άσκηση 2:

Θεωρήστε την παρακάτω ακολουθία εντολών MIPS:

```
lw  $3, 0($6)
lw  $4, 4($6)
add $8, $3, $4
sw  $8, 0($7)
```

[1] Με την ερμηνεία που δίνουμε, οι διαδικασιακές εξαρτήσεις καλύπτουν και άλματα χωρίς συνθήκη, παρ' όλο που η ροή του κώδικα δεν περνάει ποτέ από τις εντολές που ακολουθούν τέτοια άλματα. Ο κίνδυνος οφείλεται στο γεγονός ότι μέχρι την εκτέλεση των αλμάτων, οι αμέσως επόμενες εντολές μπορούν να ξεκινήσουν τον κύκλο τους χωρίς να πρέπει.

[2] Αν θέλαμε μεγαλύτερη ακρίβεια στην ανάλυσή μας, θα έπρεπε να θεωρήσουμε την πιθανότητα η μία από τις δύο εντολές που ακολουθούν μια εντολή μνήμης να είναι εντολή άλματος, οπότε τα δύο παγώματα της τελευταίας καλύπτουν το πάγωμα της πρώτης.

Οι εντολές αυτές εκτελούνται σε αρχιτεκτονική μερικά επικαλυπτόμενων εντολών, όπου ο κύκλος εντολής είναι χωρισμένος σε 5 φάσεις, με τον τρόπο που είδαμε στο μάθημα.
Δώστε το διάγραμμα χρονισμού για την εκτέλεση των 4 εντολών όταν έχουμε (α) μια κοινή και (β) διαχωρισμένη κρυφή μνήμη εντολών και δεδομένων.
Αντιμετωπίστε τυχόν κινδύνους εξαρτήσεων από δεδομένα ως εξής:
(i) Πάγωμα της εξαρτημένης εντολής μέχρι τα δεδομένα να έχουν αποθηκευτεί στους αντίστοιχους καταχωρητές, οπότε και διαβάζονται, υποθέτοντας ότι η ανάγνωση μπορεί να γίνει αμέσως μετά την αποθήκευση και στον ίδιο κύκλο μηχανής.
(ii) Ιδανική αντιμετώπιση (με παροχέτευση), οπότε η εξαρτημένη εντολή προχωράει στη φάση εκτέλεσης, αμέσως μόλις τα δεδομένα παραχθούν.

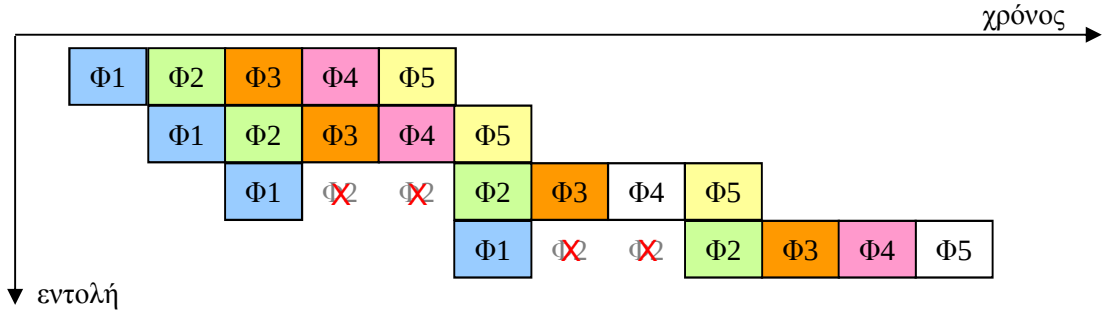
Απάντηση:

Όπως βλέπουμε στην ακολουθία των εντολών που μας δίνεται, υπάρχει εξάρτηση από δεδομένα μεταξύ της τρίτης και των δύο πρώτων, καθώς και μεταξύ της τελευταίας και της τρίτης εντολής. Ειδικότερα, οι δύο πρώτες εντολές διαβάζουν δεδομένα από τη μνήμη στους καταχωρητές $3 και $4, ενώ η τρίτη εντολή χρησιμοποιεί τα δεδομένα αυτά. Παρόμοια, η τέταρτη εντολή χρειάζεται το περιεχόμενο του καταχωρητή $8, ο οποίος γράφεται από την προηγούμενη εντολή.

Πέρα από εξαρτήσεις από δεδομένα, η δεδομένη ακολουθία εντολών δεν εμφανίζει διαδικασιακές εξαρτήσεις, εφ' όσον δεν περιλαμβάνει εντολές άλματος.
Από την άλλη μεριά, όσο αφορά δομικές εξαρτήσεις, η μόνη τέτοια εξάρτηση που παρουσιάζεται είναι η εξάρτηση από την κρυφή μνήμη, κι αυτό συμβαίνει μόνο εφ' όσον έχουμε μία κοινή κρυφή μνήμη εντολών και δεδομένων. Στην περίπτωση αυτή, μια εντολή που χρησιμοποιεί την κρυφή μνήμη στη φάση προσπέλασης στη μνήμη αναγκάζει να παγώσει την προς ανάκληση εντολή μέχρι να ολοκληρώσει την προσπέλαση αυτή.

(i) Στη χειρότερη περίπτωση αντιμετώπισης κινδύνων εξαρτήσεων από δεδομένα η μονάδα ελέγχου παγώνει τη φάση αποκωδικοποίησης της εξαρτημένης εντολής, στην οποία διαβάζεται ο φάκελος καταχωρητών. Την ολοκληρώνει όταν η τελευταία – και επομένως και κάθε αρχαιότερη – εντολή από την οποία αυτή εξαρτάται έχει αποθηκεύσει στο φάκελο καταχωρητών όποιο δεδομένο χρειάζεται η εξαρτημένη εντολή[3].
Έτσι, για τις δεδομένες εντολές, η τρίτη εντολή δε μπορεί να ολοκληρώσει τη φάση αποκωδικοποίησης, πριν οι δύο πρώτες να έχουν ολοκληρωθεί, έχοντας αποθηκεύσει τα δεδομένα στους αντίστοιχους καταχωρητές. Αντίστοιχα, η τελευταία εντολή δε μπορεί να ολοκληρώσει τη φάση αποκωδικοποίησης, πριν η τρίτη εντολή αποθηκεύσει το αποτέλεσμα της πρόσθεσης στο φάκελο καταχωρητών.

Υποθέτουμε ότι η προσπέλαση στην κρυφή μνήμη γίνεται σε έναν κύκλο μηχανής. Διαφορετικά θα διαχωρίζαμε όποια φάση προσπελαύνει την κρυφή μνήμη σε μερικά επικαλυπτόμενες φάσεις.
Για την περίπτωση (α) μιας κοινής κρυφής μνήμης εντολών και δεδομένων έχουμε:



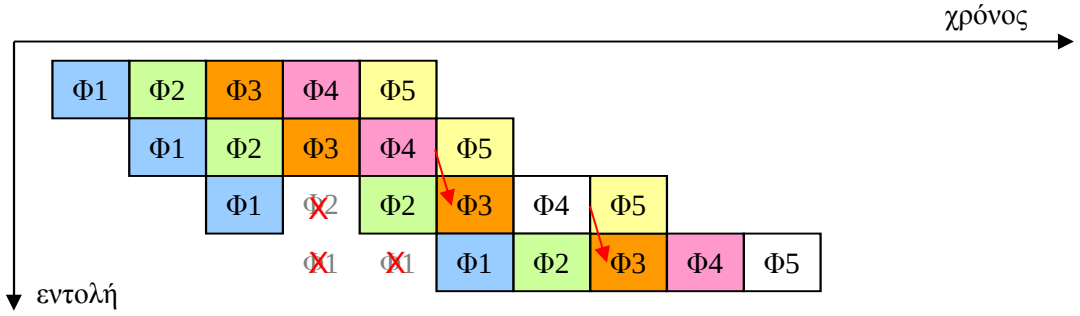
[3] Αν δεν υποθέταμε ότι η ανάγνωση μπορεί να ακολουθήσει την εγγραφή του ΦΚ στον ίδιο κύκλο μηχανής, αλλά υποθέταμε ότι κάθε προσπέλαση του ΦΚ απαιτεί έναν ολόκληρο κύκλο μηχανής, τότε θα είχαμε ένα περισσότερο πάγωμα ανά κίνδυνο.

όπου με λευκό τετραγωνάκι αναπαριστώνται οι φάσεις που δεν έχουν χρήσιμο αποτέλεσμα για την εντολή, ενώ με Χ σημειώνονται τα παγώματα στον κύκλο εντολής.

Οι κίνδυνοι εξαρτήσεων από δεδομένα της ακολουθίας εντολών που έχουμε παγώνουν τις εντολές με τέτοιον τρόπο, ώστε τις στιγμές που οι δύο πρώτες εντολές διαβάζουν τη μνήμη δεδομένων (φάση Φ4), καμία επόμενη εντολή δεν είναι στη φάση ανάκλησης (Φ1). Έτσι, το γεγονός ότι έχουμε κοινή και όχι διαχωρισμένη κρυφή μνήμη δεν έχει καμία επίδραση στην απόδοση επικάλυψης για τις 4 εντολές, και επομένως το διάγραμμα χρονισμού για την περίπτωση (β) της διαχωρισμένης κρυφής μνήμης εντολών και δεδομένων θα είναι το ίδιο με το παραπάνω, όταν έχουμε τη χειρότερη αντιμετώπιση κινδύνων εξαρτήσεων από δεδομένα.

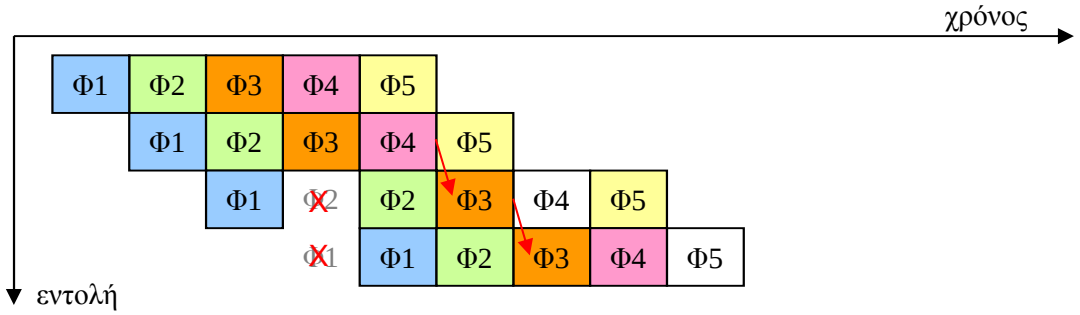
(ii) Παρατηρούμε στο παραπάνω διάγραμμα ότι οι κίνδυνοι εξαρτήσεων από δεδομένα παγώνουν την εκτέλεση εντολών για αρκετούς κύκλους μηχανής, κι επομένως ρίχνουν σημαντικά την απόδοση της τεχνικής μερικά επικαλυπτόμενων εντολών. Ευτυχώς, υπάρχουν τεχνικές που αντιμετωπίζουν αυτούς τους κινδύνους με τέτοιον τρόπο, ώστε το πάγωμα εντολών να ελαχιστοποιείται.
Στην ιδανική αντιμετώπιση κινδύνων εξαρτήσεων από δεδομένα, μια εξαρτημένη εντολή προχωράει στη φάση που χρησιμοποιεί τα δεδομένα, αμέσως μόλις αυτά γίνουν διαθέσιμα. Αυτό επιτυγχάνεται με προώθηση των δεδομένων αυτών στις υπομονάδες της ΜΕΔ που τα χρειάζονται, με την τεχνική της παροχέτευσης.
Έτσι, στην περίπτωσή μας, η τρίτη εντολή θα προχωρήσει στη φάση εκτέλεσης αμέσως μετά την ολοκλήρωση των φάσεων προσπέλασης στη μνήμη των δύο προηγούμενων εντολών. Επομένως, θα πρέπει να παγώσει, μέχρι αυτές να ολοκληρώσουν την προσπέλαση στη μνήμη και της προωθήσουν τα δεδομένα που διάβασαν. Από την άλλη μεριά, η τέταρτη εντολή χρησιμοποιεί στη φάση προσπέλασης στη μνήμη το δεδομένο που παράγει η τρίτη στη φάση εκτέλεσης. Επειδή η φάση προσπέλασης στη μνήμη της τέταρτης ούτως ή άλλως ακολουθεί τη φάση εκτέλεσης της τρίτης εντολής, η εξάρτηση αυτή από δεδομένα δεν έχει πια καμία επίδραση στο χρονισμό των δύο εντολών.

Υποθέτοντας πάλι ότι η προσπέλαση στην κρυφή μνήμη γίνεται σε έναν κύκλο μηχανής, για την περίπτωση (α) έχουμε:



όπου με κόκκινο βελάκι δείχνουμε τις παροχετεύσεις που ενεργοποιούνται κατά την εκτέλεση του κώδικα. Παρατηρούμε ότι η δομική εξάρτηση από την κοινή κρυφή μνήμη παγώνει την τέταρτη εντολή μέχρι την ολοκλήρωση της φάσης προσπέλασης στη μνήμη (Φ4) των δύο πρώτων.

Στην περίπτωση (β) όμως έχουμε:



Στην τελευταία αυτή περίπτωση η τέταρτη εντολή ξεκινάει στο συντομότερο δυνατό χρόνο, δηλαδή 3 κύκλους μετά την πρώτη εντολή. Το μόνο πάγωμα που έχουμε είναι της τρίτης εντολής, η οποία πρέπει λόγω του παραμένοντος κινδύνου λόγω εξάρτησης από δεδομένα να περιμένει την ολοκλήρωση της φάσης προσπέλασης στη μνήμη (Φ4) της δεύτερης εντολής. Η φάση εκτέλεσης (Φ3) της τρίτης εντολής θα πρέπει έτσι να περιμένει έναν κύκλο μηχανής.

Παρατηρούμε ότι στη χειρότερη αντιμετώπιση κινδύνων εξαρτήσεων από δεδομένα ο συνολικός χρόνος ολοκλήρωσης των 4 εντολών είναι 12 κύκλοι μηχανής (συνυπολογίζοντας και τη φάση Φ5 της τελευταίας εντολής παρόλο που δεν κάνει τίποτε χρήσιμο), ο οποίος είναι 6 κύκλους μικρότερος από το σειριακό χρόνο εκτέλεσης (4×5-2 = 18 κύκλοι μηχανής) και 6 κύκλους μεγαλύτερος από τον ιδανικό χρόνο εκτέλεσης (5+3 = 8 κύκλοι μηχανής)[4].

Οι τεχνικές που οδηγούν σε ελαχιστοποίηση παγώματος για εξαρτήσεις από δεδομένα μηδενίζουν το χαμένο χρόνο για κινδύνους από εντολές ΑΛΜ και μειώνουν σε έναν κύκλο παγώματος κινδύνους από εντολές ανάγνωσης μνήμης. Έτσι, ο συνολικός χρόνος ολοκλήρωσης των 4 εντολών – για την περίπτωση διαχωρισμένης κρυφής μνήμης – γίνεται 9 κύκλοι μηχανής, ο οποίος είναι μόλις 1 κύκλο μηχανής μεγαλύτερος από τον ιδανικό.
Αξίζει να σημειωθεί τέλος, ότι ο αριθμός κύκλων παγώματος σε κινδύνους εξαρτήσεων από δεδομένα μπορεί να μηδενιστεί τελείως, εάν ο μεταγλωττιστής παρεμβάλλει μεταξύ εντολών που εμφανίζουν τέτοια εξάρτηση άλλες ανεξάρτητες εντολές. Έτσι η φάση στην οποία η εξαρτημένη εντολή χρησιμοποιεί τα δεδομένα τα οποία περιμένει απομακρύνεται χρονικά από τη φάση στην οποία αυτά παράγονται. Με κατάλληλο αριθμό παρεμβαλλόμενων εντολών κάθε εξαρτημένη εντολή μπορεί να προχωρήσει κανονικά χωρίς να χρειαστεί να περιμένει για κανέναν κύκλο μηχανής.

Άσκηση 3:

Μελετήστε τις εξαρτήσεις στον πιο κάτω κώδικα MIPS:

```
addi  $6, $0, 1
ori   $13, $12, 0xffff
and   $4, $13, $6
addi  $9, $9, 4
lw    $6, -4($9)
beq   $6, $8, -5
sub   $7, $4, $5
```

τόσο από δεδομένα, όσο και διαδικασιακές.
Εάν δεν έχουμε μηχανισμό παροχέτευσης στη ΜΕΔ, δείξτε τους κινδύνους που εμφανίζονται από τις εξαρτήσεις δεδομένων.
Υποθέστε για τη συνέχεια ότι η ΜΕΔ διαθέτει μηχανισμό παροχέτευσης δεδομένων προς την ΑΛΜ. Δείξτε ποιοι κίνδυνοι εξαρτήσεων από δεδομένα δε μπορούν να αντιμετωπιστούν και μας αναγκάζουν να παγώσουμε το μηχανισμό μερικής επικάλυψης.
Με πρόβλεψη μη εκτέλεσης άλματος, σχεδιάστε το διάγραμμα χρονισμού για την εκτέλεση των πιο πάνω εντολών, (α) όταν το άλμα δε γίνεται και (β) όταν το άλμα γίνεται.
Υποθέστε τώρα ότι έχετε καθυστέρηση ενός κύκλου μηχανής στην εκτέλεση του άλματος. Αναδιατάξτε τις εντολές στον πιο πάνω κώδικα, ώστε (i) να καλύψετε τη θέση καθυστέρησης χωρίς να χρειαστεί να εισάγετε τη ψευδοεντολή nop, και (ii) να εξαλείψετε τον παραμένοντα κίνδυνο από δεδομένα.
Επανασχεδιάστε τα διαγράμματα χρονισμού. Τι παρατηρείτε;

[4] Υπενθυμίζουμε ότι στη σειριακή εκτέλεση πολλαπλών κύκλων μηχανής οι φάσεις που δεν κάνουν τίποτε χρήσιμο παραλείπονται. Για τις εντολές που έχουμε εδώ, δεν θα εκτελούνται ούτε η φάση Φ4 της τρίτης ούτε η φάση Φ5 της τέταρτης εντολής. Στην αρχιτεκτονική μερικά επικαλυπτόμενων εντολών όμως, καμία φάση δεν παραλείπεται, και όλες οι εντολές περνάνε από όλες οι φάσεις. Παρόλο που οι συγκεκριμένες φάσεις συνεχίζουν να μην κάνουν τίποτε χρήσιμο, δεν παραλείπονται, οπότε για τον υπολογισμό του χρονισμού της επικάλυψης πρέπει να λάβουμε υπ' όψη όλες τις φάσεις.

Απάντηση:

Οι εξαρτήσεις δεδομένων σε έναν κώδικα δημιουργούνται λόγω των εγγραφών δεδομένων σε καταχωρητές ή στη μνήμη, αφού γενικά ό,τι γράφεται από κάποια εντολή, διαβάζεται ή επανεγγράφεται σε κάποια άλλη εντολή, είτε επόμενη είτε προηγούμενη, κι έτσι η σχετική θέση των δύο εντολών είναι κρίσιμη για την ορθότητα του κώδικα. Οι εξαρτήσεις από δεδομένα κατηγοριοποιούνται σε τρεις τύπους, ανάλογα με τη σχετική θέση των εμπλεκόμενων εντολών: (i) Ανάγνωσης Μετά από Εγγραφή (AME), (ii) Εγγραφής Μετά από Ανάγνωση (EMA) και (iii) Εγγραφής Μετά από Εγγραφή (EME).

Στον κώδικα που μας δίνεται εντοπίζουμε εξαρτήσεις δεδομένων που αφορούν τους καταχωρητές $4, $6, $9 και $13. Πιο συγκεκριμένα, αριθμώντας τις εντολές από Ε1 έως Ε7 με τη σειρά που μας δίνονται, οι εξαρτήσεις τύπου AME που έχουμε είναι: (α) Ε1 → Ε3, λόγω ανάγνωσης μετά από εγγραφή του $6, (β) Ε1 → Ε6, λόγω ανάγνωσης μετά από εγγραφή του $6, (γ) Ε2 → Ε3, λόγω ανάγνωσης μετά από εγγραφή του $13, (δ) Ε3 → Ε7, λόγω ανάγνωσης μετά από εγγραφή του $4, (ε) Ε4 → Ε5, λόγω ανάγνωσης μετά από εγγραφή του $9, και (στ) Ε5 → Ε6, λόγω ανάγνωσης μετά από εγγραφή του $6.

Στις παραπάνω εξαρτήσεις τύπου AME πρέπει να προσθέσουμε και τις εξαρτήσεις που δημιουργούνται λόγω της εντολής διακλάδωσης Ε6, η οποία επειδή έχει αρνητική μετατόπιση, δημιουργεί δομή βρόχου στον κώδικά μας. Αυτές οι εξαρτήσεις που μεταφέρονται μέσω του βρόχου είναι: (ζ) $E4^{(i)} \rightarrow E4^{(i+1)}$, λόγω ανάγνωσης μετά από εγγραφή του $9, και (η) $E5^{(i)} \rightarrow E3^{(i+1)}$, λόγω ανάγνωσης μετά από εγγραφή του $6.

Οι εξαρτήσεις τύπου EMA που έχουμε στον παραπάνω κώδικα είναι: (θ) Ε3 → Ε5, λόγω εγγραφής μετά από ανάγνωση του $6, (ι) Ε4 → Ε4, λόγω εγγραφής μετά από ανάγνωση του $9, και οι μεταφερόμενες μέσω του βρόχου: (ια) $E3^{(i)} \rightarrow E2^{(i+1)}$, λόγω εγγραφής μετά από ανάγνωση του $13, (ιβ) $E5^{(i)} \rightarrow E4^{(i+1)}$, λόγω εγγραφής μετά από ανάγνωση του $9, και (ιγ) $E6^{(i)} \rightarrow E5^{(i+1)}$, λόγω εγγραφής μετά από ανάγνωση του $6.

Οι εξαρτήσεις τύπου EME που έχουμε στον παραπάνω κώδικα είναι: (ιδ) Ε1 → Ε5, λόγω εγγραφής μετά από εγγραφή του $6, και οι μεταφερόμενες μέσω του βρόχου: (ιε) $E2^{(i)} \rightarrow E2^{(i+1)}$, λόγω εγγραφής μετά από εγγραφή του $13, (ιστ) $E3^{(i)} \rightarrow E3^{(i+1)}$, λόγω εγγραφής μετά από εγγραφή του $4, (ιζ) $E4^{(i)} \rightarrow E4^{(i+1)}$, λόγω εγγραφής μετά από εγγραφή του $9, και (ιη) $E5^{(i)} \rightarrow E5^{(i+1)}$, λόγω εγγραφής μετά από εγγραφή του $6.

Όλες οι εξαρτήσεις που μεταφέρονται μέσω του βρόχου δείχνονται να υπάρχουν από την επανάληψη i προς την επανάληψη i+1, όμως θεωρητικά υπάρχουν και προς όλες τις επόμενες επαναλήψεις.

Επίσης, αξίζει να σημειωθεί ότι πολλές εξαρτήσεις είναι ακίνδυνες, διότι εμφανίζονται μεταξύ εντολών όπου άλλες εξαρτήσεις έχουν ήδη επιβάλει τη σωστή σειρά εκτέλεσής τους. Για παράδειγμα, η εξάρτηση (β) είναι ακίνδυνη, εφόσον υπάρχουν οι εξαρτήσεις (ιδ) και (στ) που ούτως ή άλλως αναγκάζουν την Ε6 να μη μπορεί να διαβάσει τα δεδομένα που γράφει η Ε1. Το ίδιο θα μπορούσαμε να πούμε και για εξαρτήσεις που μεταφέρονται μέσω του βρόχου για απόσταση μεγαλύτερης της μίας επανάληψης.

Όλα τα παραπάνω μπορούν να φανούν καθαρά σε έναν πίνακα όπου παρουσιάζονται οι αναγνώσεις (Α) και οι εγγραφές (Ε) όλων των καταχωρητών που συμβαίνουν σε όλες τις εντολές του παραπάνω κώδικα, για τους καταχωρητές που γράφονται τουλάχιστον μία φορά, και για εκτέλεση δύο επαναλήψεων του βρόχου. Όλες οι εξαρτήσεις προκύπτουν κοιτάζοντας τις διαδοχικές αναγνώσεις και εγγραφές των καταχωρητών. Επιπλέον, με τον πίνακα αυτόν γίνεται φανερό ότι δε χρειάζεται να μελετήσουμε περισσότερες των δύο επαναλήψεων, και πιο συγκεκριμένα ότι δε χρειάζεται να λάβουμε υπόψη εξαρτήσεις που εκτείνονται πέρα από δύο διαδοχικές εγγραφές του ίδιου καταχωρητή από την ίδια εντολή διαδοχικών επαναλήψεων, όπως για παράδειγμα την εξάρτηση $E2^{(i)} \rightarrow E3^{(i+1)}$.

Επιπλέον των εξαρτήσεων από δεδομένα, ο κώδικας εμφανίζει και μια διαδικασιακή εξάρτηση, την Ε6 → Ε7, λόγω της εντολής διακλάδωσης Ε6. Παρόλο που η Ε7 τελικά θα εκτελεστεί στην έξοδο από το βρόχο, στο εύρος μίας επανάληψης δε μπορούμε να γνωρίζουμε αν η Ε7 θα εκτελεστεί πριν την εκτέλεση της εντολής διακλάδωσης Ε6.

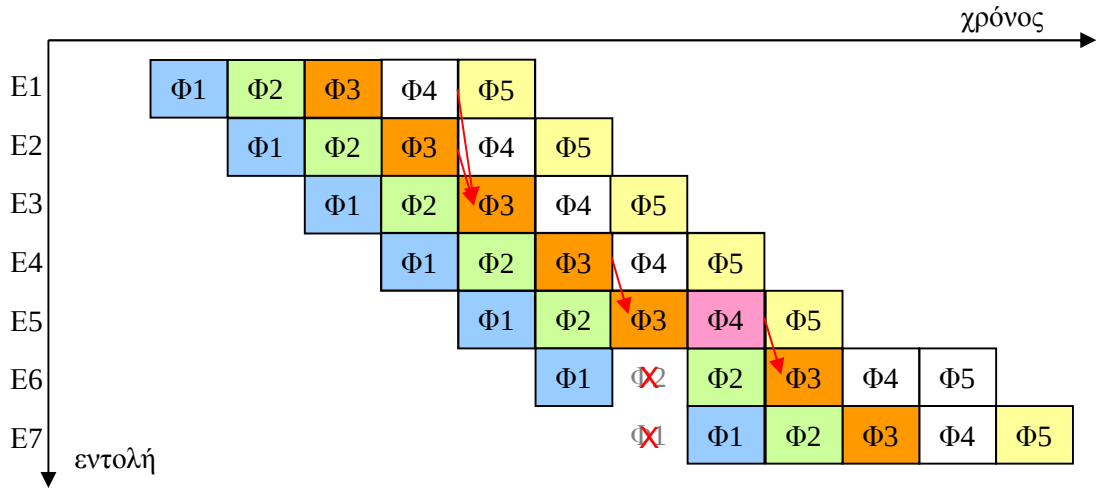
Εντολές	Καταχωρητές				
	$6	$13	$4	$9	$7
E1	E				
$E2^{(i)}$		E			
$E3^{(i)}$	A	A	E		
$E4^{(i)}$				A/E	
$E5^{(i)}$	E			A	
$E6^{(i)}$	A				
$E2^{(i+1)}$		E			
$E3^{(i+1)}$	A	A	E		
$E4^{(i+1)}$				A/E	
$E5^{(i+1)}$	E			A	
$E6^{(i+1)}$	A				
E7			A		E

Στην αρχιτεκτονική MIPS με μερική επικάλυψη που μελετήσαμε στο μάθημα, οι εντολές εκτελούνται σειριακά, με τη φάση αποθήκευσης αποτελέσματος να βρίσκεται πάντα μετά τη φάση ανάγνωσης τελούμενων. Έτσι, οι εξαρτήσεις δεδομένων τύπου ΕΜΑ, παρόλο που υπάρχουν θεωρητικά, δε δημιουργούν ποτέ κίνδυνο, αφού κάθε εγγραφή γίνεται μετά τις αναγνώσεις όλων των προηγούμενων εντολών. Παρόμοια, και εφόσον οι εντολές ολοκληρώνονται με την ίδια σειρά που προσκομίζονται, ούτε οι εξαρτήσεις τύπου ΕΜΕ δε δημιουργούν κίνδυνο, αφού κάθε εγγραφή γίνεται μετά τις εγγραφές όλων των προηγούμενων εντολών. Επομένως, από όλες τις παραπάνω εξαρτήσεις δεδομένων, οι μόνες που μας απασχολούν για μια ΜΕΔ MIPS είναι οι εξαρτήσεις τύπου ΑΜΕ (α) – (η).
Από τις παραπάνω εξαρτήσεις (α) – (η) κίνδυνο σε μια ΜΕΔ MIPS δημιουργούν μόνο όσες αναφέρονται σε εντολές που είναι τόσο κοντά μεταξύ τους, ώστε να υπάρχει ενδεχόμενο η δεύτερη εντολή να διαβάζει τα τελούμενά της πριν τα γράψει η πρώτη. Αν δεν υπάρχει κάποιος μηχανισμός παροχέτευσης, και εφόσον ο ΦΚ μπορεί να διαβαστεί στον ίδιο κύκλο που συμβαίνει κάποια εγγραφή, κίνδυνος δημιουργείται όταν η δεύτερη εντολή είναι η επόμενη ή η μεθεπόμενη της πρώτης, και όχι κατ' ανάγκη στη στατική σειρά των εντολών του κώδικα, αφού λόγω των διακλαδώσεων οι ακολουθίες των εντολών καθορίζονται δυναμικά. Αν η εξαρτημένη εντολή είναι πιο μακριά, τότε θα μπορέσει να διαβάσει τα σωστά δεδομένα από το ΦΚ. Έτσι, οι εξαρτήσεις δεδομένων του κώδικά μας που δημιουργούν κίνδυνο σε μια ΜΕΔ MIPS χωρίς παροχέτευση θα είναι οι (α), (γ), (ε) και (στ). Αυτές οι εξαρτήσεις θα οδηγούν σε πάγωμα της ΜΕΔ μέχρι η ανάγνωση των τελούμενων της εξαρτημένης εντολής να συμβαίνει στον ίδιο κύκλο με την εγγραφή των δεδομένων που αυτή περιμένει.

Αν η ΜΕΔ στην οποία εκτελείται ο κώδικάς μας περιλαμβάνει μηχανισμό παροχέτευσης δεδομένων για αντιμετώπιση κινδύνων εξαρτήσεων από δεδομένα, κάθε κίνδυνος από δεδομένα αντιμετωπίζεται επιτυχώς με αποφυγή παγώματος του μηχανισμού μερικής επικάλυψης της ΜΕΔ, για όλες τις περιπτώσεις που οι εμπλεκόμενες εντολές παράγουν και χρησιμοποιούν δεδομένα μόνο στην ΑΛΜ. Για τις περιπτώσεις που η παραγωγή των δεδομένων γίνεται στη μνήμη δεδομένων, το οποίο σημαίνει ότι η εντολή που παράγει τα δεδομένα είναι εντολή φόρτωσης, η παροχέτευση αντιμετωπίζει τον κίνδυνο, μόνο αν η εξαρτημένη εντολή *δεν* είναι η αμέσως επόμενη της εντολής φόρτωσης. Διαφορετικά, τα δεδομένα δεν έχουν προλάβει να παραχθούν και η χρήση τους από την εξαρτημένη εντολή είναι αδύνατη. Επομένως, σε μια ΜΕΔ MIPS μερικής επικάλυψης με μηχανισμό παροχέτευση προς την ΑΛΜ, οι μόνες εξαρτήσεις δεδομένων που αποτελούν κίνδυνο είναι οι εξαρτήσεις τύπου ΑΜΕ από εντολή φόρτωσης προς την αμέσως επόμενη εντολή.
Από τις εξαρτήσεις δεδομένων που έχουμε στον κώδικά μας, μόνο οι (στ) και (η) είναι εξαρτήσεις από τη μνήμη δεδομένων, από τις οποίες η (η) εξ αρχής δεν αποτελούσε κίνδυνο. Όμως η εξάρτηση (στ) είναι προς την αμέσως επόμενη εντολή της εντολής φόρτωσης (E5 → E6), κι επομένως αποτελεί τον μόνο παραμένοντα κίνδυνο από τον κώδικά μας. Για το λόγο αυτό, στην εντολή E6, η χρήση των δεδομένων που αναφέρονται στον καταχωρητή $6 δε

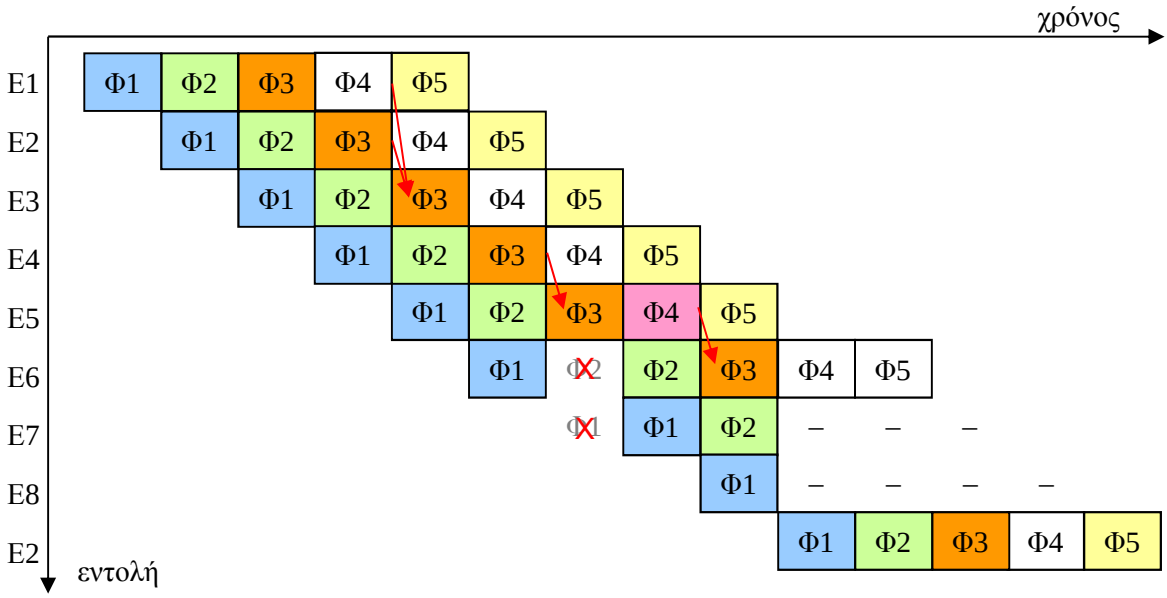
μπορεί να γίνει πριν το τέλος της φάσης Φ4 προσπέλασης της μνήμης δεδομένων της προηγούμενης εντολής Ε5, η οποία είναι εντολή που διαβάζει μια λέξη δεδομένων από τη μνήμη. Επειδή οι Ε5 και Ε6 είναι διαδοχικές εντολές, ο μηχανισμός παροχέτευσης δεν αρκεί για την αντιμετώπιση του κινδύνου από τη μεταξύ τους εξάρτηση, και η εντολή Ε6 θα πρέπει να παγώσει για έναν κύκλο μηχανής, σε αναμονή των δεδομένων από τη μνήμη. Ο μηχανισμός παροχέτευσης θα προωθήσει τα δεδομένα στην ΑΛΜ για τη συνέχιση της εντολής Ε6 στον επόμενο κύκλο μηχανής.

Με επιτυχή πρόβλεψη μη εκτέλεσης του άλματος, ο χρονισμός των εντολών του κώδικα δίνεται στο ακόλουθο διάγραμμα:



όπου με βελάκι σημειώνονται οι γραμμές παροχέτευσης που ενεργοποιούνται, ενώ με λευκό τετραγωνάκι αναπαριστώνται οι φάσεις που δεν έχουν χρήσιμο αποτέλεσμα για την εντολή. Διακρίνουμε το πάγωμα της Ε6 (και ως εκ τούτου και της Ε7) κατά έναν κύκλο μηχανής.

Σε περίπτωση αποτυχημένης πρόβλεψης, το διάγραμμα χρονισμού γίνεται ως εξής:



όπου Ε8 είναι η εντολή που ακολουθεί την Ε7. Βλέπουμε ότι το κόστος της λανθασμένης πρόβλεψης είναι δύο κύκλοι μηχανής, οι οποίοι προστίθενται στον έναν κύκλο από το πάγωμα της Ε6, για να πάρουμε συνολικά τρεις χαμένους κύκλους στο χρονισμό των εντολών από μία εκτέλεση του δεδομένου κώδικα. Παρατηρούμε ακόμα ότι λόγω της εκτέλεσης της εντολής διακλάδωσης στη Φ3, η εντολή Ε8 που ακολουθεί την Ε7 έχει ήδη αρχίσει τον κύκλο της, πριν η αποτίμηση της συνθήκης άλματος διαγράψει τόσο αυτήν όσο και την Ε7, που στο μεταξύ έχει μπει στη φάση Φ2.

Μέχρι τώρα υποθέσαμε ότι τα άλματα της αρχιτεκτονικής μας εκτελούνται χωρίς καθυστέρηση. Στη συνέχεια θα υποθέσουμε ότι τα άλματα εκτελούνται με καθυστέρηση ενός κύκλου[5], δηλαδή μετά την εντολή που είναι αμέσως κάτω από την εντολή άλματος. Με καθυστέρηση ενός κύκλου, η εντολή αυτή εκτελείται πάντα, ακόμα και αν η εκτέλεση του άλματος αλλάζει τη ροή του κώδικα, κάτι που συμβαίνει σε κάθε περίπτωση άμεσων ή έμμεσων αλμάτων, αλλά και στην περίπτωση επιτυχημένων διακλαδώσεων.
Ο αρχικός κώδικας μας δίνεται χωρίς κάποια υπόθεση καθυστέρησης στην εκτέλεση της εντολής διακλάδωσης E6. Αυτό σημαίνει ότι τόσο στην περίπτωση αλμάτων χωρίς καθυστέρηση, όσο και στην περίπτωση αλμάτων με καθυστέρηση, η εντολή E7 που ακολουθεί την εντολή διακλάδωσης E6 εξαρτάται από αυτήν, και δεν εκτελείται όταν η διακλάδωση είναι επιτυχημένη και το άλμα προς την E2 εκτελείται. Στην περίπτωση αλμάτων με καθυστέρηση, όμως, για σωστή εκτέλεση του κώδικα όσο αφορά την εντολή διακλάδωσης E6[6], θα πρέπει να συμπληρώσουμε τη θέση καθυστέρησης της εντολής αυτής. Αυτό γίνεται, για να τοποθετήσουμε στη θέση καθυστέρησης της E6 μια εντολή που δεν εξαρτάται από αυτήν, και η οποία στην απλούστερη περίπτωση μπορεί να είναι η ψευδοεντολή nop.
Ακόμα καλύτερα, η εντολή στη θέση καθυστέρησης μπορεί να είναι μια από τις εντολές που προηγούνται, ώστε να αποφύγουμε την πρόσθεση μιας κατά τα άλλα άχρηστης εντολής. Μια τέτοια μετακίνηση όμως δεν είναι εύκολη, διότι πρέπει να προσέξουμε, ώστε να μην παραβιαστεί καμία από τις εξαρτήσεις που έχουμε στον κώδικα. Μάλιστα, στην περίπτωση αυτή δεν κοιτάζουμε μόνο τις εξαρτήσεις AME, αλλά όλες τις εξαρτήσεις που προαναφέραμε, τόσο τις EMA όσο και τις EME, μια που τώρα δεν μιλάμε για κινδύνους που εμφανίζονται κατά την εκτέλεση της συγκεκριμένης σειράς εντολών, αλλά για αναδιάταξη του κώδικα που οδηγεί σε νέα σειρά εκτέλεσης των εντολών.
Η εντολή που θα μετακινήσουμε για να την τοποθετήσουμε στη θέση καθυστέρησης δε μπορεί να είναι η E5, η οποία λόγω της εξάρτησης (στ) τύπου AME πρέπει να εκτελείται πριν την E6. Μα τότε δε μπορεί να είναι ούτε η E4, η οποία λόγω της εξάρτησης (ε) τύπου AME πρέπει να εκτελείται πριν την E5. Επίσης όμως, δε μπορεί να είναι η E3, εφ' όσον τότε εμπλέκεται η εξάρτηση (θ) τύπου EMA, σύμφωνα με την οποία η E3 πρέπει να εκτελείται πριν την E5. Η E2 λόγω της εξάρτησης (γ) τύπου AME πρέπει να εκτελείται πριν την E3. Τέλος, και η E1 πρέπει να εκτελείται πριν την E2, λόγω της εξάρτησης (α) τύπου AME. Όλα αυτά βέβαια οδηγούν στο συμπέρασμα ότι φαινομενικά δεν υπάρχει καμία εντολή η οποία να μπορεί να τοποθετηθεί στη θέση καθυστέρησης της E6.
Παρά το παραπάνω συμπέρασμα όμως, υπάρχουν κάποια τρικ, τα οποία μας επιτρέπουν να πετύχουμε το στόχο μας. Ένα τέτοιο τρικ σχετίζεται με συγκεκριμένη ακολουθία εντολών που αφορά τις αυξομειώσεις κατά σταθερά της τιμής του καταχωρητή βάσης μιας ή περισσότερων εντολών προσπέλασης μνήμης. Πιο συγκεκριμένα, αν σε έναν κώδικα βρίσκονται οι εντολές

```
addi  $a, $a, x
lw    $b, y($a)
```

για κάποιους καταχωρητές $a και $b, και κάποιες σταθερές x και y, τότε οι εντολές αυτές αναδιατάσσονται ως

```
lw    $b, z($a)
addi  $a, $a, x
```

με τη σταθερά z να ισούται με y+x. Εφόσον αναβάλλουμε την αλλαγή της τιμής του $a, πρέπει να συνυπολογίσουμε την αλλαγή στη μετατόπιση της εντολής φόρτωσης.
Με την παραπάνω τεχνική, μπορούμε πλέον στη θέση καθυστέρησης της E6 να τοποθετήσουμε την E4, τροποποιώντας τη σταθερά μετατόπισης της E5.

---

[5] Παραδοσιακά στην αρχιτεκτονική MIPS τα άλματα υλοποιούνται με έναν κύκλο καθυστέρησης. Γι' αυτό και η σχετική διευθυνσιοδότηση στις εντολές διακλάδωσης γίνεται με βάση την επόμενη της εντολής διακλάδωσης.

[6] Λανθασμένη όσο αφορά μια εντολή διακλάδωσης είναι η εκτέλεση στην οποία μία τουλάχιστον εντολή εκτελείται, ενώ δε βρίσκεται στην κατεύθυνση εκτέλεσης και επομένως δε θα έπρεπε να εκτελεστεί, ή αντίστροφα δεν εκτελείται, ενώ βρίσκεται στην κατεύθυνση εκτέλεσης και επομένως θα έπρεπε να εκτελεστεί.

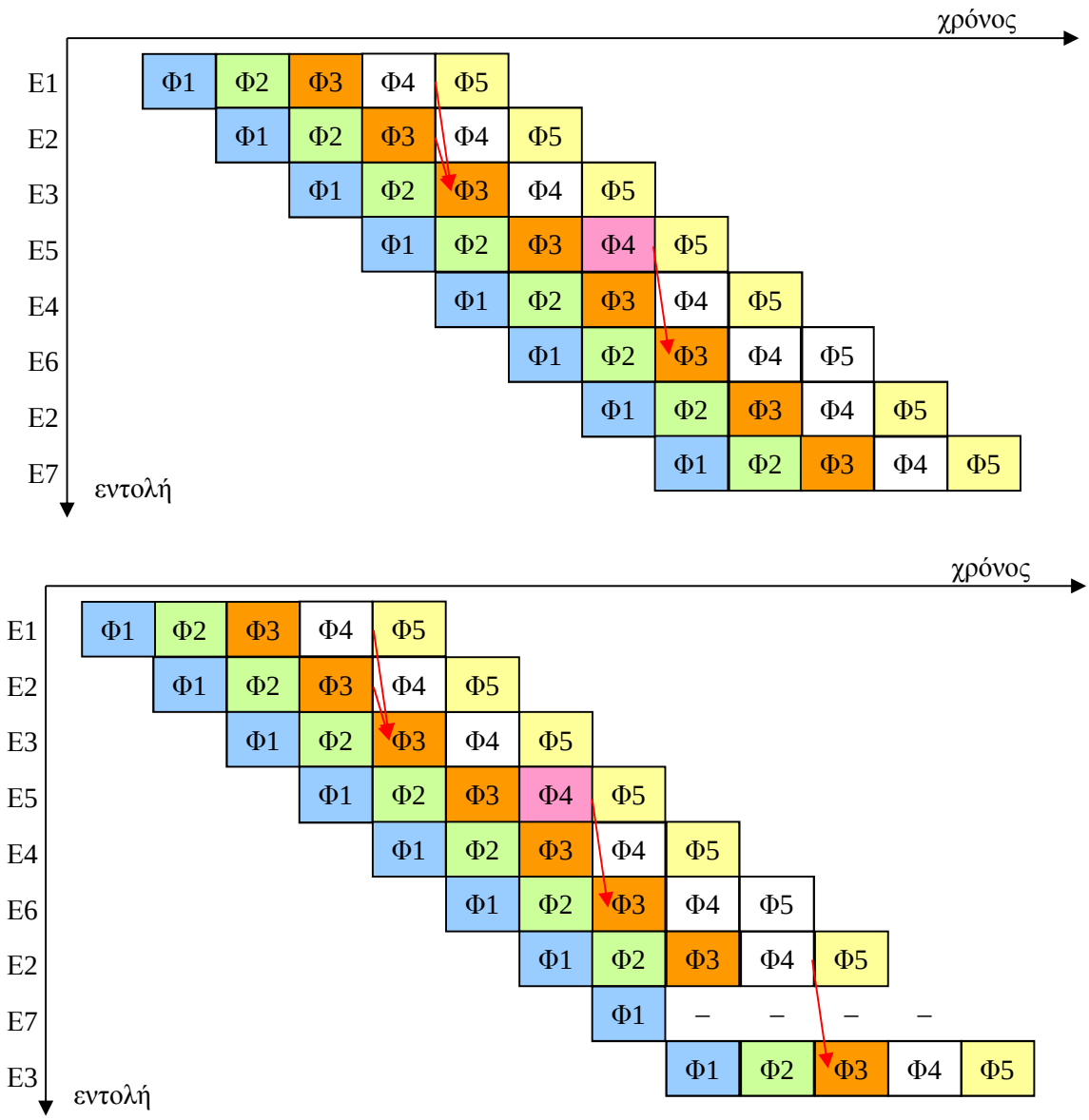
Ένα δεύτερο τρικ που μπορούμε να εφαρμόσουμε είναι η αντιγραφή υπό προϋποθέσεις της εντολής προορισμού του άλματος στη θέση καθυστέρησης, με ταυτόχρονη αλλαγή στον προορισμό, ώστε ο προορισμός τώρα να είναι μια εντολή πιο κάτω. Η βασική προϋπόθεση μιας τέτοιας αναδιάταξης είναι η εξασφάλιση ορθότητας του κώδικα, δεδομένου ότι η εντολή που αντιγράψαμε δε θα πρέπει να εκτελείται αν το άλμα δεν εκτελείται. Προφανώς δεν υπάρχει πρόβλημα για άμεσα ή έμμεσα άλματα, αφού τότε το άλμα πάντα εκτελείται. Πρόβλημα υπάρχει όμως για διακλαδώσεις, όπου θα πρέπει να εξασφαλίσουμε ότι το αποτέλεσμα της λανθασμένα εκτελεσμένης εντολής δεν επηρεάζει την ορθότητα του κώδικα.
Στην περίπτωσή μας, το ζητούμενο είναι η αντιγραφή της εντολής E2 στη θέση καθυστέρησης της E6. Αρκεί να διερευνήσουμε κατά πόσο η εντολή E2 βλάπτει την ορθότητα του κώδικα στην έξοδο από το βρόχο. Παρατηρούμε ότι το αποτέλεσμα της εντολής E2 δεν εξαρτάται από την εκτέλεση του βρόχου. Μ' άλλα λόγια, η τιμή που λαμβάνει ο καταχωρητής $13 είναι η ίδια σε κάθε επανάληψη του βρόχου. Άρα, μια ακόμα εκτέλεση της E2 στο τέλος του βρόχου δεν αλλοιώνει τις τιμές που γράφονται στο ΦΚ από τις εντολές του βρόχου, οπότε η αντιγραφή της E2 στη θέση καθυστέρησης της E6 είναι επιτρεπτή. Ας σημειωθεί, ότι ακόμα κι αν η τιμή του καταχωρητή $13 προέκυπτε διαφορετική σε κάθε επανάληψη, θα ήταν δυνατή η αντιγραφή, αν δεν υπάρχει εξάρτηση τύπου ΑΜΕ από την E2 προς εντολή που ακολουθεί το βρόχο, ή, ακόμα κι αν υπάρχει τέτοια εξάρτηση, αν είναι εφικτό να βρούμε και να προσθέσουμε μία ακόμα εντολή που να αναιρεί το αποτέλεσμα της λανθασμένα εκτελεσμένης E2 στην έξοδο από το βρόχο. Για μεγάλο αριθμό επαναλήψεων, οι προσθήκες που κάνουμε προσθέτουν αμελητέο κόστος.

Φυσικά για την συμπλήρωση της θέσης καθυστέρησης της εντολής διακλάδωσης E6, μας αρκεί ένα από τα δύο τρικ που αναφέραμε. Όμως, η αναδιάταξη των εντολών αποσκοπεί επιπλέον και στη μείωση των χαμένων κύκλων, πέρα από εκείνους που χάνονται στην εκτέλεση μιας εντολής διακλάδωσης. Στην περίπτωσή μας, τέτοιος χαμένος κύκλος είναι ο κύκλος παγώματος της E6, λόγω αναμονής δεδομένων από τη μνήμη. Για να καλύψουμε λοιπόν το χρόνο αναμονής που δημιουργεί το πρόβλημα στην εξάρτηση (στ), θα πρέπει να βάλουμε κάποια εντολή μεταξύ των E5 και E6. Έτσι, με το δεύτερο από τα παραπάνω τρικ πετυχαίνουμε και αυτό το στόχο.

Ο τελικός κώδικας που παίρνουμε μετά τη μικρή αναδιάταξη που κάναμε είναι ο εξής:

```
E1:   addi  $6, $0, 1
E2:   ori   $13, $12, 0xffff
E3:   and   $4, $13, $6
E5:   lw    $6, 0($9)
E4:   addi  $9, $9, 4
E6:   beq   $6, $8, -4
E2:   ori   $13, $12, 0xffff
E7:   sub   $7, $4, $5
```

όπου η μετατόπιση της εντολής διακλάδωσης έχει διορθωθεί κατάλληλα, ώστε να οδηγεί στην πρώτη εντολή του σώματος του βρόχου, που τώρα είναι η E3.
Σε περίπτωση επιτυχούς πρόβλεψης μη εκτέλεσης άλματος στην E6, το διάγραμμα χρονισμού είναι αυτό που δίνεται στην αρχή της επόμενης σελίδας. Παρατηρούμε ότι με τη νέα διάταξη του κώδικα, επιτυγχάνουμε στην περίπτωση αυτή το βέλτιστο χρόνο στην εκτέλεση των εντολών. Στην περίπτωση αποτυχημένης πρόβλεψης το διάγραμμα χρονισμού ακολουθεί το προηγούμενο. Παρατηρούμε όμως τώρα ότι δεν καταφέρνουμε να εξαλείψουμε πλήρως τους χαμένους κύκλους από την εντολή διακλάδωσης E6. Η εισαγωγή της θέσης καθυστέρησης μείωσε τους χαμένους κύκλους που είχαμε σε έναν, αλλά δεν οδήγησε σε βέλτιστη εκτέλεση. Η καθυστέρηση ενός κύκλου μηδενίζει το κόστος της λανθασμένης πρόβλεψης, μόνο εάν η εντολή εκτελείται στη φάση Φ2. Στην περίπτωσή μας, ακόμα και αν η αρχιτεκτονική υποστήριζε κάτι τέτοιο, ο συνδυασμός της εντολής διακλάδωσης E6 με την εντολή E5 από την οποία αυτή εξαρτάται, αποτρέπει την εκτέλεση της πρώτης στην Φ2, μια που ακόμα και μετά την αναδιάταξη το δεδομένο δεν είναι διαθέσιμο στη Φ2 της E6. Έτσι, για τον κώδικά μας θα θέλαμε μία ακόμα εντολή να παρεμβληθεί μεταξύ των E5 και E6, ώστε να εξαλείψουμε πλήρως την απώλεια κύκλων από τη λανθασμένη πρόβλεψη της E6.



Παρατηρήστε ότι στον παραπάνω αναδιαταγμένο κώδικα οι εξαρτήσεις δεδομένων δεν είναι πια αυτές που είχαμε βρει για τον αρχικό κώδικα. Για παράδειγμα, η ΑΜΕ εξάρτηση (ε) μεταξύ των Ε4 και Ε5 έχει μετατραπεί σε εξάρτηση που μεταφέρεται μέσω του βρόχου, γι' αυτό και δεν υπάρχει η παροχέτευση που είχαμε νωρίτερα μεταξύ των δύο εντολών. Αντίθετα, η ΕΜΑ εξάρτηση (ιβ) που μεταφερόταν μέσω του βρόχου τώρα αναφέρεται στην ίδια επανάληψη. Επίσης, η ΑΜΕ εξάρτηση (γ) μεταξύ των Ε2 και Ε3 συμπληρώνεται και με μια νέα εξάρτηση τύπου ΑΜΕ από το αντίγραφο της Ε2 που μεταφέρεται μέσω του βρόχου στην επόμενη Ε3. Η τελευταία μάλιστα ενεργοποιεί και παροχέτευση, όπως φαίνεται στο πιο πάνω διάγραμμα, εφόσον η εξαρτημένη εντολή Ε3 είναι η μεθεπόμενη της Ε2 της προηγούμενης επανάληψης στη δυναμική σειρά εκτέλεσης των εντολών του κώδικα!

Γενικά το πρόβλημα της διάταξης ή δρομολόγησης εντολών (instruction scheduling) για βέλτιστη εκτέλεση κώδικα είναι πολύ σημαντικό πρόβλημα της επιστήμης των υπολογιστών, και απαντάται σε πολλές περιοχές της (πχ αρχιτεκτονική, μεταγλωττιστές, προγραμματισμός, παράλληλη επεξεργασία). Όλοι οι μοντέρνοι επεξεργαστές υλοποιούν μηχανισμό μερικής αναδιάταξης εντολών στο υλικό τους, αλλά η προεργασία που κάνει ο μεταγλωττιστής στην αρχική διάταξη των εντολών του κώδικα είναι το κλειδί για την αποτελεσματικότερη λειτουργία κάθε επεξεργαστή.

Άσκηση 4:

Η επιτυχία πρόβλεψης ενός μηχανισμού πρόβλεψης διακλάδωσης ορίζεται σαν το λόγο του αριθμού επιτυχών προβλέψεων προς το συνολικό αριθμό προβλέψεων του μηχανισμού.
Μελετήστε την επιτυχία πρόβλεψης ενός μηχανισμού (α) στατικής πρόβλεψης στηριγμένου στο πρόσημο της μετατόπισης, (β) δυναμικής πρόβλεψης με 1 ψηφίο ιστορίας, και (γ) δυναμικής πρόβλεψης με 2 ψηφία ιστορίας ανά διακλάδωση, με ολίσθηση του νεώτερου και επιλογή του αρχαιότερου ψηφίου για την πρόβλεψη, στις ακόλουθες περιπτώσεις αλμάτων διακλαδώσεων σε βρόχο Ν επαναλήψεων:

1. Άλμα τερματισμού βρόχου προς την αρχή του βρόχου.
2. Άλμα με σταθερή κατεύθυνση που αλλάζει στο μισό του αριθμού επαναλήψεων του βρόχου.
3. Άλμα προς τα εμπρός που εκτελείται μεμονωμένα στο 10% των επαναλήψεων του βρόχου.
4. Άλμα προς τα εμπρός που εκτελείται κάθε δεύτερη επανάληψη
5. Άλμα προς τα εμπρός που εκτελείται κάθε τρίτη επανάληψη.
6. Άλμα προς τα εμπρός που δεν *εκτελείται κάθε τρίτη επανάληψη.*

Μπορείτε να εντοπίσετε εσείς κάποια περίπτωση στην οποία ο τρίτος από τους παραπάνω μηχανισμούς αποτυγχάνει τελείως; Σχολιάστε.

Απάντηση:

Στη στατική πρόβλεψη στηριγμένη στο πρόσημο της μετατόπισης της εντολής διακλάδωσης, προβλέπουμε ότι το άλμα εκτελείται, όταν η μετατόπιση είναι αρνητική. Μια τέτοια πρόβλεψη αποσκοπεί στη βέλτιστη επικάλυψη εντολών σε βρόχους, όταν αυτοί τερματίζονται με εντολή διακλάδωσης προς την αρχή του βρόχου, και συνεπώς το άλμα της εντολής αυτής εκτελείται τις περισσότερες φορές. Από την άλλη μεριά, διακλαδώσεις με θετική μετατόπιση αντιμετωπίζονται σαν περιπτώσεις αλμάτων που δεν εκτελούνται συχνά, κι επομένως η πρόβλεψη γι' αυτές είναι μη εκτέλεση του άλματος.
Στη δυναμική πρόβλεψη, διατηρούμε έναν αριθμό ψηφίων ιστορίας ανά διακλάδωση, τα οποία κωδικοποιούν τις πιο πρόσφατες αποτιμήσεις της συνθήκης άλματος της διακλάδωσης. Αληθής συνθήκη (δηλαδή εκτέλεση του άλματος) αντιστοιχεί σε ψηφίο 1, ψευδής συνθήκη (δηλαδή μη εκτέλεση του άλματος) αντιστοιχεί σε ψηφίο 0. Η πρόβλεψη υπολογίζεται από τα ψηφία ιστορίας με βάση κάποιον αλγόριθμο, που συνήθως είναι απλά η πλειοψηφία των τιμών τους. Έτσι, περισσότερα 1 οδηγούν σε πρόβλεψη εκτέλεσης του άλματος, και αντίστροφα, περισσότερα 0 οδηγούν σε πρόβλεψη μη εκτέλεσης του άλματος. Για άρτιο αριθμό ψηφίων ιστορίας, η περίπτωση ίσου αριθμού 1 και 0 αντιμετωπίζεται ιδιαίτερα. Στην άσκηση αυτή, για τη δυναμική πρόβλεψη με 2 ψηφία ιστορίας ανά διακλάδωση, θα θεωρήσουμε την τεχνική στην οποία η ενημέρωση της ιστορίας σε μια διακλάδωση γίνεται με ολίσθηση σε αυτήν του νέου ψηφίου που προκύπτει από τη διακλάδωση, ενώ η πρόβλεψη γίνεται με επιλογή της αρχαιότερης τιμής κατεύθυνσης άλματος.

Η αρχική πρόβλεψη στην περίπτωση δυναμικής πρόβλεψης, όταν δηλαδή δεν έχουμε ιστορία για τη διακλάδωση στον πίνακα ιστορίας διακλαδώσεων, γίνεται σε πολλές περιπτώσεις με στατική πρόβλεψη. Για να μελετήσουμε τη συμπεριφορά της δυναμικής πρόβλεψης χωρίς την επίδραση άλλου μηχανισμού στην αρχική πρόβλεψη, θα υποθέσουμε ότι ο βρόχος των Ν επαναλήψεων είναι μέρος ενός μεγαλύτερου εξωτερικού βρόχου, και ότι έχει ήδη εκτελεστεί τουλάχιστον μια φορά μέσα σε αυτόν τον εξωτερικό βρόχο.

Για κάθε περίπτωση διακλάδωσης έχουμε[7]:

1. *Άλμα προς τα πίσω στο τέλος του βρόχου.*
Η στατική πρόβλεψη επιτυγχάνει σε κάθε εκτέλεση της διακλάδωσης, εκτός από την τελευταία, στην οποία το άλμα δεν εκτελείται. Η επιτυχία πρόβλεψης E_σ θα είναι:

[7] Για να μην αναλωθούμε σε μελέτες ειδικών περιπτώσεων, που είναι πέρα από τους σκοπούς της άσκησης, θα υποθέσουμε ότι ο Ν έχει κατάλληλη τιμή για κάθε περίπτωση, πχ. άρτιος, πολλαπλάσιο του 3, μεγαλύτερος του 10, κλπ.

$E_\sigma = (N-1)/N$

Η δυναμική πρόβλεψη με ένα ψηφίο ιστορίας αποτυγχάνει δύο φορές στην εκτέλεση του βρόχου, την πρώτη φορά που η πρόβλεψη είναι «μη εκτέλεση» και το άλμα εκτελείται, και την τελευταία φορά που η πρόβλεψη είναι «εκτέλεση» και το άλμα δεν εκτελείται. Η πρώτη πρόβλεψη γίνεται επειδή στην αμέσως προηγούμενη εκτέλεση της εντολής διακλάδωσης, δηλαδή στην τελευταία της προηγούμενης εκτέλεσης του βρόχου, το άλμα δεν εκτελέστηκε. Η επιτυχία πρόβλεψης $E_{\delta 1}$ στην περίπτωση αυτή θα είναι:

$E_{\delta 1} = (N-2)/N$

Η δυναμική πρόβλεψη με δύο ψηφία ιστορίας μπορεί να προβλέψει την πρώτη εκτέλεση του άλματος. Αυτό συμβαίνει, επειδή η τελευταία εκτέλεση του άλματος σε μια εκτέλεση του βρόχου, ακολουθούμενη από τη μη εκτέλεση στην έξοδο από αυτόν, δίνουν τιμή 10 στα ψηφία ιστορίας. Αυτή η τιμή οδηγεί ξανά σε πρόβλεψη εκτέλεσης στην επόμενη εμφάνιση της εντολής διακλάδωσης, που θα είναι η πρώτη επανάληψη στην επόμενη εκτέλεση του βρόχου. Όμως, η εκτέλεση της δεύτερης επανάληψης δε θα προβλεφθεί, επειδή τα ψηφία ιστορίας έχουν τώρα τιμή 01, γεγονός που οδηγεί σε πρόβλεψη μη εκτέλεσης. Έτσι, η επιτυχία πρόβλεψης $E_{\delta 2}$ γίνεται και πάλι:

$E_{\delta 2} = (N-2)/N$

Παρατηρούμε ότι σε αυτή την κατηγορία διακλάδωσης και για μεγάλο N, και οι τρεις μηχανισμοί έχουν παραπλήσια απόδοση, η οποία είναι πολύ κοντά στο 100%. Για χαμηλές τιμές του N όμως, η δυναμική πρόβλεψη υστερεί έναντι της στατικής.

2. *Σταθερή κατεύθυνση άλματος με μία αλλαγή στη μέση του βρόχου.* Υποθέτουμε ότι η επόμενη εκτέλεση του βρόχου εμφανίζει την ίδια ακριβώς ιστορία στη διακλάδωση που μας απασχολεί.

Η στατική πρόβλεψη στην περίπτωση αυτή αποτυγχάνει στις μισές εκτελέσεις της διακλάδωσης, ανεξάρτητα από το πρόσημο της μετατόπισης. Έτσι, η επιτυχία πρόβλεψης E_σ είναι μόλις:

$E_\sigma = 50\%$

Η δυναμική πρόβλεψη με ένα ψηφίο ιστορίας αποτυγχάνει δύο φορές σε αυτή την περίπτωση, μια φορά στην αλλαγή κατεύθυνσης άλματος στη μέση του βρόχου και μια φορά στην αρχή του βρόχου, όταν η διακλάδωση επανέρχεται στην ιστορία της προηγούμενης εκτέλεσης του βρόχου. Άρα η επιτυχία $E_{\delta 1}$ είναι:

$E_{\delta 1} = (N-2)/N$

Η δυναμική πρόβλεψη με δύο ψηφία ιστορίας από την άλλη μεριά, αποτυγχάνει τέσσερις φορές. Σε κάθε αλλαγή κατεύθυνσης άλματος, είτε στη μέση του βρόχου, είτε στην αρχή της επόμενης εκτέλεσής του, κάνει δύο φορές λάθος, επειδή τα ψηφία ιστορίας χρειάζονται δύο εκτελέσεις της διακλάδωσης, για να μπορέσουν να προβλέψουν σωστά τη νέα κατεύθυνση. Για παράδειγμα, τα ψηφία ιστορίας 00 στην αλλαγή κατεύθυνσης άλματος θα κάνουν το πρώτο λάθος και θα πάρουν τιμή 01. Όμως η τιμή 01 προβλέπει την αρχαιότερη κατεύθυνση, δηλαδή «μη εκτέλεση», η οποία είναι και πάλι λανθασμένη. Στη συνέχεια η τιμή των ψηφίων ιστορίας γίνεται 11, η οποία θα προβλέψει σωστά «εκτέλεση» στην επόμενη εμφάνιση της διακλάδωσης, και θα συνεχίσει να προβλέπει σωστά μέχρι την επόμενη αλλαγή κατεύθυνσης. Συνολικά:

$E_{\delta 2} = (N-4)/N$

Παρατηρούμε τώρα ότι η στατική πρόβλεψη αποτυγχάνει σε αυτή την κατηγορία διακλάδωσης, μια που η επιτυχία πρόβλεψης είναι πολύ χαμηλή σε σύγκριση με τους άλλους δύο μηχανισμούς. Οι μηχανισμοί δυναμικής πρόβλεψης έχουν υψηλή επιτυχία πρόβλεψης, που για μεγάλα N πλησιάζει στο 100%. Για μικρότερα N ο μηχανισμός δυναμικής πρόβλεψης με ένα ψηφίο ιστορίας έχει καλύτερη απόδοση από αυτόν με δύο ψηφία ιστορίας, επειδή ο δεύτερος καθυστερεί σε σχέση με τον πρώτο στην αναγνώριση της αλλαγής κατεύθυνσης άλματος κατά μια εκτέλεση της διακλάδωσης.

3. *Άλμα προς τα εμπρός που εκτελείται μεμονωμένα στο 10% του N.*

Η στατική πρόβλεψη σε άλματα προς τα εμπρός – με θετική δηλαδή μετατόπιση – επιτυγχάνει στην πρόβλεψη, μόνο όταν το άλμα δεν εκτελείται. Επομένως, η επιτυχία πρόβλεψης θα είναι:

$E_{\sigma} = 90\%$

Η δυναμική πρόβλεψη με ένα ψηφίο ιστορίας δε μπορεί να προβλέψει μεμονωμένες εξαιρέσεις στη κατεύθυνση άλματος, και κάνει δύο λάθη για κάθε εκτέλεση άλματος, ένα στην πρόβλεψη της κατεύθυνσης όταν το άλμα εκτελείται, κι ένα στην αμέσως επόμενη εκτέλεση της εντολής διακλάδωσης, στην οποία το άλμα δεν εκτελείται, αλλά το ψηφίο ιστορίας έχει τιμή 1. Δηλαδή:

$E_{\delta 1} = 80\%$

Η δυναμική πρόβλεψη με δύο ψηφία ιστορίας παρουσιάζει παρόμοια συμπεριφορά, με τη διαφορά ότι η δεύτερη αποτυχία εμφανίζεται αργότερα. Έτσι, η επιλογή της αρχαιότερης κατεύθυνσης σε περίπτωση ψηφίων ιστορίας με τιμή 01 οδηγεί σε επιτυχία, αφού μετά την πρώτη αποτυχημένη πρόβλεψη που είναι αναπόφευκτη, η πρόβλεψη που ακολουθεί είναι «μη εκτέλεση» και είναι σωστή. Όμως, επειδή η τεχνική ενημέρωσης ιστορίας με ολίσθηση θα μας δώσει 10 ως νέα ιστορία, η επόμενη πρόβλεψη θα είναι «εκτέλεση» και είναι λανθασμένη, όταν οι μεμονωμένες εκτελέσεις του άλματος δε συμβαίνουν ποτέ σε απόσταση μικρότερη από τρεις επαναλήψεις του βρόχου. Επομένως η επιτυχία πρόβλεψης θα είναι πάλι:

$E_{\delta 2} = 80\%$

Στην περίπτωση αυτής της κατηγορίας διακλαδώσεων η στατική πρόβλεψη είναι καλή και αποτυγχάνει στον ελάχιστο αριθμό επαναλήψεων, που δε μπορεί να είναι μικρότερος από 10%. Η δυναμική πρόβλεψη δε μπορεί να αντιμετωπίσει αυτές τις διακλαδώσεις με το βέλτιστο τρόπο, αφού αποτυγχάνει δύο φορές για κάθε περίπτωση εκτέλεσης άλματος.

4. *Άλμα προς τα εμπρός που εκτελείται κάθε δεύτερη επανάληψη.*

Η στατική πρόβλεψη στην περίπτωση αυτή συμπεριφέρεται όπως στην περίπτωση (2), εφ' όσον η επιτυχία πρόβλεψης εξαρτάται μόνο από τον αριθμό των αλμάτων που εκτελούνται. Έτσι:

$E_{\sigma} = 50\%$

Η δυναμική πρόβλεψη με ένα ψηφίο ιστορίας αποτυγχάνει σε *κάθε* εκτέλεση της διακλάδωσης, επειδή η κατεύθυνση άλματος αλλάζει συνεχώς, και το ψηφίο ιστορίας περιέχει κάθε φορά την τιμή που αντιστοιχεί στην προηγούμενη εκτέλεση της διακλάδωσης. Έτσι:

$E_{\delta 1} = 0\%$

Η δυναμική πρόβλεψη με δύο ψηφία ιστορίας και με ολίσθηση στην ενημέρωση της ιστορίας προβλέπει κάθε φορά την προηγούμενη κατεύθυνση άλματος στην εκτέλεση της διακλάδωσης, η οποία είναι και η σωστή. Για παράδειγμα, μετά από διαδοχή κατευθύνσεων «μη εκτέλεση»-«εκτέλεση», τα ψηφία ιστορίας έχουν τιμή 01, η οποία οδηγεί σε πρόβλεψη μη εκτέλεσης του άλματος, που λόγω της εναλλαγής της κατεύθυνσης άλματος, είναι η σωστή. Άρα:

$E_{\delta 2} = 100\%$

Παρατηρούμε ότι στην περίπτωση αυτή ο τρίτος από τους πιο πάνω μηχανισμούς έχει απόλυτη επιτυχία. Αυτό συμβαίνει επειδή η εναλλαγή κατεύθυνσης άλματος είναι συμπεριφορά που μπορεί να προβλεφθεί, όταν ο πίνακας ιστορίας διακλαδώσεων διατηρεί τουλάχιστον δύο ψηφία ιστορίας για κάθε διακλάδωση, τα οποία ενημερώνονται με ολίσθηση, και όπου η πρόβλεψη γίνεται με βάση το αρχαιότερο ψηφίο. Με ένα ψηφίο ιστορίας η συμπεριφορά αυτή δε μπορεί να προβλεφθεί, γι' αυτό και η αποτυχία στην πρόβλεψη. Η στατική πρόβλεψη δε μπορεί να προβλέψει τη συμπεριφορά της διακλάδωσης, αλλά επιτυγχάνει στις μισές προβλέψεις, όταν το άλμα δεν εκτελείται.

5. *Άλμα προς τα εμπρός που εκτελείται κάθε τρίτη επανάληψη.*

Στην περίπτωση αυτή, η στατική πρόβλεψη επιτυγχάνει όταν το άλμα δεν εκτελείται, το οποίο συμβαίνει στα δύο τρίτα των επαναλήψεων:

$E_{\sigma} = 67\%$

Η δυναμική πρόβλεψη με ένα ψηφίο ιστορίας επιτυγχάνει μόνο όταν η κατεύθυνση άλματος δεν αλλάζει, το οποίο συμβαίνει μια φορά κάθε τρεις επαναλήψεις. Η ιστορία της διακλάδωσης έχει τη μορφή …001001001001…. Έτσι, διατηρώντας μόνο ένα ψηφίο στον πίνακα ιστορίας διακλαδώσεων γι' αυτή τη διακλάδωση, έχουμε επιτυχή πρόβλεψη μόνο όταν το ψηφίο ιστορίας είναι 0 και το άλμα δεν εκτελείται, δηλαδή μια στις τρεις φορές. Άρα:

$E_{\delta 1} = 33\%$

Όταν έχουμε δύο ψηφία ιστορίας, οποιαδήποτε τιμή για τα δύο ψηφία από τις 00 και 10 οδηγεί σε λανθασμένη πρόβλεψη μη εκτέλεσης και εκτέλεσης του άλματος αντίστοιχα, ενώ μόνο η τιμή 01 οδηγεί σε σωστή πρόβλεψη μη εκτέλεσης του άλματος. Μ' άλλα λόγια, η πρόβλεψη είναι και σε αυτό το μηχανισμό επιτυχής στο ένα τρίτο μόνο των επαναλήψεων:

$E_{\delta 2} = 33\%$

Σ' αυτή την περίπτωση ο μηχανισμός δυναμικής πρόβλεψης δεν είναι ιδιαίτερα επιτυχής, επειδή οι εναλλαγές κατεύθυνσης άλματος έχουν μορφή που δύσκολα προβλέπεται. Ακόμα και με δύο ψηφία ιστορίας αποτυγχάνουμε στην πρόβλεψη, επειδή η εναλλαγή κατεύθυνσης κάθε τρεις επαναλήψεις απαιτεί τρία τουλάχιστον ψηφία για τη σωστή πρόβλεψή της. Η στατική πρόβλεψη δίνει καλύτερο αποτέλεσμα, επειδή έχουμε πιο λίγες εκτελέσεις από μη εκτελέσεις του άλματος.

6. *Άλμα προς τα εμπρός που* δεν *εκτελείται κάθε τρίτη επανάληψη.*

Όπως και προηγουμένως, η στατική πρόβλεψη επιτυγχάνει όταν το άλμα δεν εκτελείται, το οποίο τώρα συμβαίνει μόνο στο ένα τρίτο των επαναλήψεων:

$E_{\sigma} = 33\%$

Η δυναμική πρόβλεψη με ένα ψηφίο ιστορίας παρουσιάζει παρόμοια συμπεριφορά με προηγουμένως, παρ' όλο που η ιστορία της διακλάδωσης έχει τώρα τη μορφή …110110110110…. Με ένα μόνο ψηφίο στον πίνακα ιστορίας διακλαδώσεων, έχουμε επιτυχή πρόβλεψη μόνο όταν το ψηφίο ιστορίας είναι 1 και το άλμα εκτελείται, δηλαδή πάλι μια στις τρεις φορές. Άρα:

$E_{\delta 1} = 33\%$

Η ομοιότητα συμπεριφοράς με την προηγούμενη περίπτωση υπάρχει και στο μηχανισμό των δύο ψηφίων ιστορίας. Οποιαδήποτε τιμή για τα δύο ψηφία από τις 01 και 11 οδηγεί σε λανθασμένη πρόβλεψη, ενώ μόνο η τιμή 10 οδηγεί σε σωστή πρόβλεψη:

$E_{\delta 2} = 33\%$

Στην περίπτωση αυτή η δυναμική πρόβλεψη παρουσιάζει την ίδια συμπεριφορά με προηγουμένως, ενώ το ίδιο αποτέλεσμα δίνει και η στατική πρόβλεψη, επειδή τώρα έχουμε πιο πολλές εκτελέσεις από μη εκτελέσεις του άλματος.

Από τη μελέτη των πιο πάνω περιπτώσεων παρατηρούμε τα εξής:

- Η επιτυχία της στατικής πρόβλεψης εξαρτάται αποκλειστικά από τον αριθμό εκτελέσεων ή μη εκτελέσεων του άλματος. Πιο πολλές εκτελέσεις σε διακλαδώσεις με αρνητική μετατόπιση και πιο λίγες σε διακλαδώσεις με θετική μετατόπιση αυξάνουν την επιτυχία πρόβλεψης.
- Η επιτυχία δυναμικών μηχανισμών πρόβλεψης εξαρτάται κυρίως από τη μορφή της ιστορίας των διακλαδώσεων. Οι μελέτες που έχουν γίνει για διάφορους μηχανισμούς επιχειρούν να βελτιστοποιήσουν την απόδοσή τους για τις πιο συνηθισμένες μορφές ιστορίας διακλαδώσεων. Έτσι διακλαδώσεις των τύπων (1), (2) και (3) μπορούν να αντιμετωπιστούν επιτυχώς με δυναμική πρόβλεψη. Επίσης, διακλαδώσεις που εμφανίζουν περιοδικότητα στη συμπεριφορά τους, όπως συμβαίνει στους τύπους (4), (5) και (6), μπορούν να αντιμετωπιστούν με επιτυχία έως και 100%, ανάλογα με τον αλγόριθμο πρόβλεψης, και μόνο για κατάλληλο αριθμό ψηφίων ιστορίας[8].

Για να βρούμε μια περίπτωση πλήρους αποτυχίας της δυναμικής πρόβλεψης με δύο ψηφία ιστορίας, ενημέρωση με ολίσθηση, και επιλογή του αρχαιότερου ψηφίου για την πρόβλεψη, αρκεί να βρούμε μια συμπεριφορά διακλάδωσης που δεν ακολουθεί ποτέ την πρόβλεψη. Μ' άλλα λόγια, με ψηφία ιστορίας 00 και 01 να εκτελεί το άλμα της, ενώ με ψηφία ιστορίας 10 και 11 να μην το εκτελεί. Μια τέτοια συμπεριφορά εμφανίζεται στην περίπτωση εναλλαγής κατεύθυνσης άλματος κάθε δύο ακριβώς επαναλήψεις, με ιστορία διακλάδωσης της μορφής

[8] Μπορεί να αποδειχτεί ότι ο ελάχιστος αριθμός ψηφίων ιστορίας που απαιτείται είναι ο λογάριθμος της περιόδου της συμπεριφοράς της διακλάδωσης. Ο ελάχιστος αριθμός όμως μπορεί να είναι κατάλληλος για συγκεκριμένες, και όχι για τις πιο συνηθισμένες μορφές ιστορίας. Για παράδειγμα, η εναλλαγή που είδαμε στον τύπο (4) μπορεί να προβλεφθεί και με ένα ψηφίο ιστορίας, αρκεί ο μηχανισμός να προγραμματιστεί έτσι, ώστε με ιστορία 0 να προβλέπει «εκτέλεση», και με ιστορία 1 να προβλέπει «μη εκτέλεση», κάτι που φυσικά δεν είναι η συνηθισμένη επιλογή πρόβλεψης!

…001100110011…. Με αυτή την ιστορία σε καμία περίπτωση ο μηχανισμός δεν προβλέπει σωστά την επόμενη εκτέλεση της διακλάδωσης, και η επιτυχία είναι:

$E_{\delta 2} = 0\%$

Άσκηση 5:

Ένα πρόγραμμα συστήματος, ο μεταγλωττιστής gcc, αποτελείται από το πιο κάτω μίγμα εντολών MIPS: 22% load, 11% store, 49% ΑΛΜ, 16% άλματα με συνθήκη, και 2% άλματα χωρίς συνθήκη.
Συγκρίνετε την επίδοση για το πρόγραμμα αυτό ενός συστήματος ΜΕΔ (α) απλού κύκλου μηχανής για κάθε κύκλο εντολής, (β) πολλαπλών κύκλων μηχανής για κάθε κύκλο εντολής, και (γ) μηχανισμού μερικά επικαλυπτόμενων εντολών με παροχέτευση.
Οι χρόνοι βασικών λειτουργιών της ΜΕΔ είναι: 2ns για προσπέλαση μνήμης, 2ns για λειτουργία ΑΛΜ, και 1ns για προσπέλαση του ΦΚ.
Για την περίπτωση μερικά επικαλυπτόμενων εντολών υποθέστε ότι οι μισές εντολές load ακολουθούνται από εξαρτώμενη εντολή, ότι το κόστος λάθους πρόβλεψης άλματος με συνθήκη είναι δύο κύκλοι μηχανής, και ότι η επιτυχία πρόβλεψης είναι 75%. Τα άλματα χωρίς συνθήκη έχουν πάντα κόστος δύο κύκλων μηχανής.
Θεωρήστε για κάθε περίπτωση τη δομή της ΜΕΔ που είδαμε στο μάθημα.

Απάντηση:

Θα αναλύσουμε την επίδοση κάθε ΜΕΔ για το συγκεκριμένο πρόγραμμα, υπολογίζοντας το μέσο ρυθμό εκτέλεσης εντολών γι' αυτό το πρόγραμμα. Για το σκοπό αυτό, θα μελετήσουμε πρώτα τα αρχιτεκτονικά χαρακτηριστικά που μας δίνονται για τις τρεις διαφορετικές οργανώσεις ΜΕΔ, ώστε να υπολογίσουμε σε καθεμία το μέσο χρόνο εκτέλεσης κάθε κατηγορίας εντολών.

Για ΜΕΔ απλού κύκλου μηχανής για κάθε κύκλο εντολής, ο κύκλος μηχανής πρέπει να περιλαμβάνει όλες τις λειτουργίες που απαιτεί η πιο πολύπλοκη εντολή, η οποία είναι η εντολή ανάγνωσης από τη μνήμη. Έτσι, το μήκος του κύκλου μηχανής θα βρίσκεται από το άθροισμα του χρόνου δύο προσπελάσεων μνήμης, του χρόνου δύο προσπελάσεων του φακέλου καταχωρητών, και του χρόνου εκτέλεσης μιας πράξης ΑΛΜ.
Ο μέσος χρόνος εκτέλεσης μιας εντολής T_α οποιασδήποτε κατηγορίας θα είναι ίσος με το χρόνο του κύκλου μηχανής. Άρα:

$T_\alpha = 2 \times 2\text{ns} + 2 \times 1\text{ns} + 2\text{ns} = 8\text{ns}$

Για ΜΕΔ πολλαπλών κύκλων μηχανής για κάθε κύκλο εντολής, ο κύκλος μηχανής έχει μήκος το χρόνο της μέγιστης μικρολειτουργίας της ΜΕΔ. Αυτός είναι ο χρόνος των 2ns μιας προσπέλασης μνήμης ή εκτέλεσης μιας πράξης ΑΛΜ.
Η πιο πολύπλοκη εντολή στη ΜΕΔ αυτή, η εντολή τύπου load, χρειάζεται πέντε κύκλους μηχανής. Η εντολή τύπου store χρειάζεται τέσσερις κύκλους μηχανής, όσους και η εντολή ΑΛΜ. Άλματα με συνθήκη (branch) και άλματα χωρίς συνθήκη (jump) χρειάζονται τρεις κύκλους μηχανής. Άρα:

$T_\pi^{(\text{load})} = 5 \times 2\text{ns} = 10\text{ns}$
$T_\pi^{(\text{store})} = T_\pi^{(\text{ΑΛΜ})} = 4 \times 2\text{ns} = 8\text{ns}$
$T_\pi^{(\text{branch})} = T_\pi^{(\text{jump})} = 3 \times 2\text{ns} = 6\text{ns}$

Στη ΜΕΔ μερικά επικαλυπτόμενων εντολών ο κύκλος μηχανής είναι ο ίδιος με την προηγούμενη περίπτωση. Τώρα ο μέσος χρόνος εκτέλεσης για μια εντολή εκφράζεται ως το διάστημα από την ολοκλήρωση της προηγούμενης μέχρι την ολοκλήρωση αυτής της εντολής. Δεδομένων των ειδικών χαρακτηριστικών αυτής της οργάνωσης ΜΕΔ που μας δόθηκαν, θα βρούμε πόσο επιβαρύνονται οι εντολές κάθε κατηγορίας από κινδύνους, δηλαδή πόσο μεγαλύτερος είναι ο ζητούμενος χρόνος από έναν κύκλο μηχανής, που είναι ο ιδανικός χρόνος μεταξύ ολοκλήρωσης διαδοχικών εντολών.

Στην περίπτωση της εντολής τύπου load, εάν το αποτέλεσμα αυτής δε χρησιμοποιείται από την αμέσως επόμενη εντολή, ο μέσος χρόνος εκτέλεσης είναι ένας κύκλος μηχανής. Διαφορετικά, έχουμε πάγωμα της ΜΕΔ για έναν κύκλο μηχανής, και ο μέσος χρόνος εκτέλεσης επαυξάνεται στους δύο κύκλους μηχανής, υποθέτοντας ότι ο μηχανισμός παροχέτευσης αποτρέπει παραπάνω κύκλους παγώματος της ΜΕΔ σε εξαρτήσεις από δεδομένα. Αφού οι μισές από τις εντολές load ακολουθούνται από εξαρτώμενη εντολή, ο μέσος χρόνος εκτέλεσης όλων των εντολών αυτής της κατηγορίας είναι:

$$T_{\varepsilon}^{(load)} = 50\% \times 1 \times 2ns + 50\% \times 2 \times 2ns = 3ns$$

Στις εντολές τύπου store, όπως και στις εντολές ΑΛΜ, ο μέσος χρόνος εκτέλεσης είναι ένας κύκλος μηχανής, εφ' όσον ο μηχανισμός παροχέτευσης εξαλείφει τους κινδύνους από εξαρτήσεις δεδομένων:

$$T_{\varepsilon}^{(store)} = 1 \times 2ns = 2ns$$
$$T_{\varepsilon}^{(ΑΛΜ)} = 1 \times 2ns = 2ns$$

Στις εντολές άλματος με συνθήκη, και σε περίπτωση σωστής πρόβλεψης, ο μέσος χρόνος εκτέλεσης είναι ένας κύκλος μηχανής. Ο χρόνος επαυξάνεται κατά δύο ακόμα κύκλους μηχανής, εάν η πρόβλεψη είναι λανθασμένη. Με 75% επιτυχία στις προβλέψεις, ο μέσος χρόνος εκτέλεσης αλμάτων με συνθήκη θα είναι:

$$T_{\varepsilon}^{(branch)} = 75\% \times 1 \times 2ns + 25\% \times 3 \times 2ns = 3ns$$

ενώ για τα άλματα χωρίς συνθήκη, με επιβάρυνση δύο κύκλων μηχανής για κάθε άλμα, θα έχουμε:

$$T_{\varepsilon}^{(jump)} = 3 \times 2ns = 6ns$$

Στη συνέχεια θα υπολογίσουμε το μέσο ρυθμό εκτέλεσης εντολών για το πρόγραμμα gcc:
Για τη ΜΕΔ απλού κύκλου μηχανής αυτός θα είναι ίσος με:

$$\rho_{\alpha} = T_{\alpha}^{-1} = 0{,}125 \text{ εντολές/ns.}$$

Για τη ΜΕΔ πολλαπλών κύκλων μηχανής για κάθε κύκλο εντολής ο μέσος ρυθμός εκτέλεσης εντολών θα είναι:

$$\rho_{\pi} = (22\% \times T_{\pi}^{(load)} + 11\% \times T_{\pi}^{(store)} + 49\% \times T_{\pi}^{(ΑΛΜ)} + 16\% \times T_{\pi}^{(branch)} + 2\% \times T_{\pi}^{(jump)})^{-1} =$$
$$= 0{,}124 \text{ εντολές/ns}$$

Τέλος, για τη ΜΕΔ μερικά επικαλυπτόμενων εντολών έχουμε:

$$\rho_{\varepsilon} = (22\% \times T_{\varepsilon}^{(load)} + 11\% \times T_{\varepsilon}^{(store)} + 49\% \times T_{\varepsilon}^{(ΑΛΜ)} + 16\% \times T_{\varepsilon}^{(branch)} + 2\% \times T_{\varepsilon}^{(jump)})^{-1} =$$
$$= 0{,}407 \text{ εντολές/ns}$$

Παρατηρούμε ότι η ΜΕΔ απλού κύκλου μηχανής είναι οριακά γρηγορότερη από τη ΜΕΔ πολλαπλών κύκλων μηχανής, επειδή οι βασικές λειτουργίες της ΜΕΔ δεν έχουν τον ίδιο χρόνο, κι επομένως ο διαχωρισμός του κύκλου εντολής σε φάσεις δεν είναι ιδανικός. Έτσι, οι εντολές τύπου load έχουν επιβραδυνθεί κατά 25%, και παρ' ό,τι κάποιες άλλες εντολές έχουν επιταχυνθεί, ο μέσος ρυθμός εκτέλεσης δεν είναι υψηλότερος στη δεύτερη οργάνωση ΜΕΔ απ' ό,τι στην πρώτη.
Από την άλλη μεριά, η ΜΕΔ μερικά επικαλυπτόμενων εντολών είναι περίπου 3,27 φορές πιο γρήγορη από τις άλλες δύο. Αυτό συμβαίνει επειδή η επικάλυψη είναι σε μεγάλο βαθμό επιτυχημένη, αν και το κόστος εξάρτησης από εντολές τύπου load κατά κύριο λόγο, καθώς και το κόστος λανθασμένης πρόβλεψης σε εντολές άλματος με συνθήκη κατά δεύτερο λόγο, δεν επιτρέπουν επιτάχυνση κοντά στην ιδανική, που για πέντε φάσεις του κύκλου εντολής είναι ίση με 5.