# Πανεπιστήμιο Θεσσαλίας
## Τμήμα Πληροφορικής

*Οργάνωση Η/Υ*

Ενότητα 1η: Εισαγωγή στην Οργάνωση Η/Υ

**Άσκηση 1:**

*Αναλύστε τη διαδοχική εκτέλεση των παρακάτω εντολών MIPS με βάση τις φάσεις του κύκλου εντολής που μάθατε στην «Εισαγωγή στους Η/Υ» και επαναλάβαμε στο μάθημα:*

```
1     lw   $2, -102($7)      :     $2 = MEM[$7 - 102]
2     addi $3, $2, 201       :     $3 = $2 + 201
3     sllv $3, $3, $6        :     $3 = $3 << $6
4     lw   $4, 0($8)         :     $4 = MEM[$8]
5     add  $4, $4, $3        :     $4 = $4 + $3
6     sw   $4, 4($8)         :     MEM[$8 + 4] = $4
7     beq  $4, $10, Label    :     if $4 == $10 goto Label
```

*όπου η κύρια λειτουργία κάθε εντολής περιγράφεται με τη μορφή σχολίου δίπλα στην εντολή, και Label είναι κάποια ετικέτα στον κώδικα.*
*Στη συνέχεια σχολιάστε το χρονισμό στην εκτέλεση των εντολών χωρίς καμία επικάλυψη μεταξύ τους, το αποτέλεσμα που έχει σε αυτόν η μερική επικάλυψη των φάσεων διαδοχικών εντολών, και τις πιθανές επιπτώσεις από εκτέλεση εκτός σειράς.*
*Για τους σκοπούς της παρούσας ενότητας κάντε ελεύθερες υποθέσεις για την προσπέλαση της μνήμης.*

**Απάντηση:**

Στην «Οργάνωση Η/Υ» μελετάμε αποκλειστικά επεξεργαστές *αποθηκευμένου προγράμματος*, στους οποίους οι εντολές μηχανής είναι αποθηκευμένες στη μνήμη, απ' όπου διαβάζονται και εκτελούνται με μια σειρά λογικών φάσεων που αποτελούν τον κύκλο εντολής. Η σειρά φάσεων που μελετήσαμε στο μάθημα είναι μια βασική σειρά, της οποίας παραλλαγές υλοποιούν οι περισσότεροι επεξεργαστές: Ξεκινώντας (α) με τη φάση ανάκλησης που προσκομίζει την εντολή από τη μνήμη, συνεχίζοντας (β) με τη φάση αποκωδικοποίησης που μεταφράζει τη λέξη εντολής σε μια σειρά από σήματα ελέγχου, (γ) με τη φάση ανάγνωσης τελούμενων που φέρνει τα τελούμενα εισόδου – αν υπάρχουν – από το φάκελο καταχωρητών ή από τη μνήμη, (δ) με τη φάση εκτέλεσης, που προχωρά στην εκτέλεση της κύριας λειτουργίας της εντολής, και τελειώνοντας (ε) με τη φάση αποθήκευσης που αποθηκεύει το αποτέλεσμα – αν υπάρχει – στο φάκελο καταχωρητών ή στη μνήμη[1]. Ο κύκλος εντολής υποστηρίζεται από το Μετρητή Προγράμματος (PC), δηλαδή τον καταχωρητή που περιέχει τη διεύθυνση της επόμενης προς εκτέλεση εντολής, ο οποίος αυξάνεται αυτόματα σε κάθε εντολή, ώστε να δείχνει στην εντολή που ακολουθεί.

[1] Στην πραγματικότητα κάθε επεξεργαστής έχει το δικό του κύκλο εντολής. Ο επεξεργαστής MIPS, ως τύπου φόρτωσης-αποθήκευσης, υποστηρίζει ανάγνωση τελούμενων και αποθήκευση αποτελέσματος μόνο στο φάκελο καταχωρητών. Οι προσπελάσεις μνήμης για τις εντολές φόρτωσης (ανάγνωση μνήμης) και αποθήκευσης (εγγραφή μνήμης) υλοποιούνται σε μία πρόσθετη φάση, τη φάση προσπέλασης μνήμης, η οποία ακολουθεί τη φάση εκτέλεσης. Για τις εντολές αυτές, η φάση εκτέλεσης περιορίζεται στον υπολογισμό της τελικής διεύθυνσης προσπέλασης. Η γενική περιγραφή που ακολουθούμε, παρόλο που αναφερόμαστε σε εντολές MIPS, δε διαχωρίζει κάποια φάση προσπέλασης μνήμης, και θεωρεί την προσπέλαση μνήμης ως μέρος της εκτέλεσης, ότι δηλαδή ο υπολογισμός της διεύθυνσης προσπέλασης και η προσπέλαση της μνήμης γίνονται στη σειρά σε μια ενιαία φάση εκτέλεσης.

Έτσι λοιπόν, με βάση τις παραπάνω φάσεις του κύκλου εντολής, και για τη δεδομένη ακολουθία εντολών, ο επεξεργαστής ξεκινά με την ανάκληση της πρώτης εντολής από τη μνήμη. Στη συνέχεια αποκωδικοποιεί την εντολή που μόλις διάβασε, αναγνωρίζοντας την εντολή φόρτωσης μιας λέξης δεδομένων από τη μνήμη. Κύρια λειτουργία της εντολής αυτής είναι η προσκόμιση μιας λέξης δεδομένων από τη μνήμη, από μια διεύθυνση που καθορίζεται με βάση τα τελούμενα εισόδου της εντολής. Έτσι, στη φάση ανάγνωσης τελούμενων, ο επεξεργαστής διαβάζει τον καταχωρητή $7 από το φάκελο καταχωρητών[2]. Με την επόμενη φάση, τη φάση εκτέλεσης, ο επεξεργαστής προσθέτει το περιεχόμενο του καταχωρητή $7 με τη σταθερά -102, ώστε να υπολογίσει την τελική διεύθυνση προσπέλασης μνήμης, την οποία και να προωθήσει από τη μονάδα επεξεργασίας δεδομένων στη μονάδα διαχείρισης μνήμης για τη λειτουργία ανάγνωσης. Η μονάδα διαχείρισης μνήμης διαβάζει από τη μνήμη μια λέξη δεδομένων, την οποία και επιστρέφει στη μονάδα επεξεργασίας δεδομένων. Στην τελευταία φάση του κύκλου εντολής ο επεξεργαστής αποθηκεύει τη λέξη αυτή στον καταχωρητή $2 του φακέλου καταχωρητών.

Η δεύτερη εντολή ξεκινά με την ανάκλησή της από τη μνήμη. Με την αποκωδικοποίησή της αναγνωρίζεται η εντολή πρόσθεσης με μια σταθερά. Στη συνέχεια διαβάζεται ο καταχωρητής $2 από το φάκελο καταχωρητών, που σημειωτέον έχει μόλις πάρει νέα τιμή με την ολοκλήρωση της προηγούμενης εντολής. Στην επόμενη φάση προστίθεται η σταθερά 201 στο περιεχόμενο του καταχωρητή $2, ενώ στη φάση αποθήκευσης αποτελέσματος το αποτέλεσμα της πρόσθεσης γράφεται στον καταχωρητή $3 του φακέλου καταχωρητών.

Στη συνέχεια ανακαλείται η τρίτη εντολή από τη μνήμη, η οποία αποκωδικοποιείται με αναγνώριση εντολής αριστερής ολίσθησης με μεταβλητό αριθμό θέσεων ολίσθησης, που καθορίζεται από το δεύτερο από τα δύο τελούμενα εισόδου. Με τη φάση ανάγνωσης τελούμενων διαβάζονται οι καταχωρητές $3 και $6 από το φάκελο καταχωρητών, από τους οποίους μάλιστα ο πρώτος εγγράφηκε από την προηγούμενη εντολή. Στη συνέχεια εκτελείται η ολίσθηση, ενώ το αποτέλεσμα αποθηκεύεται πίσω στον καταχωρητή $3 του φακέλου καταχωρητών. Παρατηρήστε ότι στην εντολή αυτή ο καταχωρητής $3 αποτελεί τόσο τελούμενο εισόδου, όσο και τελούμενο εξόδου. Με σωστή διαδοχή των φάσεων του κύκλου εντολής, πρώτα διαβάζεται η παλαιά τιμή του καταχωρητή και ύστερα γράφεται η νέα του τιμή.

Η τέταρτη εντολή ξεκινά με τη φάση ανάκλησής της από τη μνήμη. Η αποκωδικοποίησή της αναγνωρίζει για άλλη μια φορά εντολή φόρτωσης από τη μνήμη. Στη συνέχεια διαβάζεται ο καταχωρητής $8 από το φάκελο καταχωρητών, το περιεχόμενο του οποίου προστίθεται με το 0 για να δώσει την τελική διεύθυνση προσπέλασης της μνήμης. Η διεύθυνση στέλνεται στη μονάδα διαχείρισης μνήμης για ανάγνωση, και η τελευταία προσκομίζει το ζητούμενο δεδομένο από τη μνήμη. Στη φάση αποθήκευσης αποτελέσματος το δεδομένο που διαβάστηκε αποθηκεύεται στον καταχωρητή $4 του φακέλου καταχωρητών.

Η πέμπτη εντολή ανακαλείται από τη μνήμη και αποκωδικοποιείται. Αναγνωρίζεται έτσι η εντολή πρόσθεσης δύο καταχωρητών. Με τη φάση ανάγνωσης τελούμενων διαβάζονται οι δύο καταχωρητές $3 και $4, οι οποίοι απ' ό,τι είδαμε, εγγράφηκαν από τις δύο προηγούμενες εντολές. Τα περιεχόμενά τους προστίθενται στη φάση εκτέλεσης και το αποτέλεσμα της πρόσθεσης αποθηκεύεται στον καταχωρητή $4 του φακέλου καταχωρητών στην τελευταία φάση του κύκλου εντολής.

Μετά την ανάκληση της έκτης εντολής ακολουθεί η αποκωδικοποίησή της για να αναγνωριστεί η εντολή αποθήκευσης στη μνήμη. Στη συνέχεια διαβάζονται τα τελούμενα από τους καταχωρητές $4 και $8 του φακέλου καταχωρητών, από τους οποίους ο $4 έχει μόλις εγγραφεί. Η φάση εκτέλεσης προσθέτει το περιεχόμενο του $8 με τη σταθερά 4 και στέλνει το αποτέλεσμα στη μονάδα διαχείρισης μνήμης σα διεύθυνση αποθήκευσης. Η τελευταία αποθηκεύει στη μνήμη το περιεχόμενο του καταχωρητή $4.

Με την τελευταία εντολή, αφού προηγηθεί η ανάκλησή της, αναγνωρίζεται άλμα υπό συνθήκη (ή αλλιώς διακλάδωση). Τα δύο τελούμενα εισόδου λαμβάνονται από τους καταχωρητές $4 και $10 του φακέλου καταχωρητών, και στη συνέχεια, στη φάση εκτέλεσης, συγκρίνονται

[2] Ας σημειώσουμε εδώ ότι η σταθερά -102, αν και είναι τελούμενο εισόδου, δε διαβάζεται ούτε από το φάκελο καταχωρητών ούτε από τη μνήμη, αλλά παρέχεται απ' ευθείας από τη λέξη εντολής, γι' αυτό και είναι ήδη διαθέσιμη από προηγούμενη φάση του κύκλου εντολής.

μεταξύ τους για ισότητα. Αν η συνθήκη ισότητας είναι αληθής, η εντολή οδηγεί σε εκτέλεση άλματος σε διεύθυνση που ορίζεται από την ετικέτα Label. Με σχετική διευθυνσιοδότηση του προορισμού άλματος απαιτείται υπολογισμός της διεύθυνσης προορισμού, κάτι που μπορεί να γίνει είτε στην ίδια μονάδα στην οποία γίνεται η σύγκριση, αλλά πριν ή μετά από αυτήν, είτε σε άλλη επιπρόσθετη μονάδα. Το άλμα ολοκληρώνεται στο τέλος της φάσης εκτέλεσης με εγγραφή της διεύθυνσης προορισμού στον PC, ακυρώνοντας έτσι την αυτόματη αύξηση του τελευταίου. Αν η συνθήκη ισότητας είναι ψευδής, δε θα εκτελεστεί άλμα και θα ακολουθήσει κανονικά η επόμενη εντολή.

Όταν έχουμε έναν επεξεργαστή στον οποίο οι εντολές εκτελούνται σειριακά, ο χρόνος ολοκλήρωσης της δεδομένης ακολουθίας εντολών είναι απλά το άθροισμα των χρόνων ολοκλήρωσης των επιμέρους εντολών. Σε μια απλοϊκή υλοποίηση επεξεργαστή αποθηκευμένου προγράμματος, κάθε εντολή ολοκληρώνεται σε μια στοιχειώδη χρονική μονάδα (ή αλλιώς έναν κύκλο μηχανής, δηλαδή έναν κύκλο του ρολογιού του επεξεργαστή), οπότε ένας κώδικας 7 εντολών θα απαιτεί 7 κύκλους μηχανής.
Σε μια πιο ρεαλιστική υλοποίηση, ο διαχωρισμός του κύκλου εντολής σε λογικές φάσεις συνοδεύεται και από αντίστοιχο διαχωρισμό του χρόνου ολοκλήρωσης μιας εντολής σε χρονικές φάσεις, πολλαπλάσια του κύκλου μηχανής του επεξεργαστή. Στην περίπτωσή μας, μπορούμε να υποθέσουμε ότι η αποκωδικοποίηση κάθε εντολής, η προσπέλαση του φακέλου καταχωρητών και η εκτέλεση πράξεων απαιτούν έναν κύκλο μηχανής. Ειδικά για το φάκελο καταχωρητών, υποθέτουμε ότι κάθε προσπέλαση για ανάγνωση τελούμενων μπορεί να διαβάσει μέχρι και δύο καταχωρητές ταυτόχρονα. Τέλος, μπορούμε να υποθέσουμε για απλούστευση ότι η προσπέλαση μνήμης ολοκληρώνεται επίσης σε έναν κύκλο μηχανής.
Σύμφωνα με τις παραπάνω υποθέσεις, οι λογικές φάσεις ανάκλησης, αποκωδικοποίησης, ανάγνωσης τελούμενων και αποθήκευσης αποτελέσματος απαιτούν από έναν κύκλο μηχανής η καθεμία. Η φάση εκτέλεσης από την άλλη μεριά απαιτεί δύο κύκλους μηχανής για εντολές προσπέλασης μνήμης, έναν για τον υπολογισμό της τελικής διεύθυνσης κι έναν για την προσπέλαση της μνήμης, και έναν κύκλο μηχανής για τις υπόλοιπες εντολές. Στο σύνολό της, κάθε εντολή απαιτεί τόσους κύκλους, όσοι είναι οι κύκλοι μηχανής των φάσεων που αυτή περιλαμβάνει. Έτσι, μια εντολή φόρτωσης απαιτεί 6 κύκλους μηχανής. Μια εντολή αποθήκευσης απαιτεί 5 κύκλους μηχανής, αφού έχει μεν φάση προσπέλασης μνήμης, αλλά δεν έχει φάση αποθήκευσης αποτελέσματος. Μια εντολή αριθμητικής/λογικής πράξης ή ολίσθησης απαιτεί 5 κύκλους μηχανής. Μια εντολή διακλάδωσης, τέλος, η οποία δεν έχει αποτέλεσμα, ούτε φυσικά προσπέλαση μνήμης, απαιτεί 4 κύκλους μηχανής, υποθέτοντας ότι ο υπολογισμός της διεύθυνσης προορισμού άλματος δεν γίνεται μετά τη σύγκριση.
Συνολικά λοιπόν, σε έναν τέτοιο σειριακό επεξεργαστή η πιο πάνω ακολουθία εντολών απαιτεί:

$$6 + 5 + 5 + 6 + 5 + 5 + 4 = 36$$

κύκλους μηχανής[3].

Αν διαθέτουμε έναν επεξεργαστή με μερική επικάλυψη των φάσεων του κύκλου εντολής, μπορεί να γίνει επικάλυψη κάθε εντολής με τις επόμενες, αρκεί να μην παραβιάζονται οι όποιες εξαρτήσεις μεταξύ των εντολών. Προϋπόθεση για επικαλυπτόμενη υλοποίηση του κύκλου εντολής είναι ο διαχωρισμός αυτού σε ισόχρονες φάσεις, με τέτοιον τρόπο, ώστε κάθε φάση να μπορεί να απασχολείται από διαφορετική εντολή από μια ακολουθία διαδοχικών εντολών, ανά κύκλο μηχανής. Κάθε εντολή πρέπει να περνάει από όλες τις φάσεις, παρόλο που σε κάποιες από αυτές μπορεί να μην κάνει τίποτα. Ιδανικά, ο χρόνος ολοκλήρωσης ν εντολών θα είναι ίσος με το χρόνο ολοκλήρωσης μιας εντολής, συν ν-1 κύκλους μηχανής που απαιτούνται για την ολοκλήρωση των υπόλοιπων ν-1 εντολών. Αυτό συμβαίνει, επειδή με τη μερική επικάλυψη κάθε εντολή ολοκληρώνεται έναν κύκλο μηχανής μετά την προηγούμενη.
Στην περίπτωσή μας θα είχαμε 6 επικαλυπτόμενες φάσεις, επειδή όπως αναφέραμε πιο πάνω, οι εντολές προσπέλασης μνήμης απαιτούν δύο κύκλους μηχανής στην εκτέλεσή τους, κι επο-

[3] Αν και πιο ρεαλιστική, η υλοποίηση αυτή δεν είναι και η πραγματική για επεξεργαστή MIPS. Σε επόμενη ενότητα θα δούμε μια υλοποίηση κύκλου εντολής MIPS πολύ πιο κοντά στην πραγματική.

μένως για να εφαρμόσουμε την τεχνική μερικής επικάλυψης, θα πρέπει να χωρίσουμε τη φάση εκτέλεσης σε δύο φάσεις του ενός κύκλου, αλλά και επειδή η εντολή φόρτωσης απαιτεί το μεγαλύτερο χρόνο από όλες τις εντολές, καθώς χρησιμοποιεί όλες τις φάσεις του κύκλου εντολής.
Έτσι, για 7 εντολές θα απαιτούνται ιδανικά:

6 + 6 = 12

κύκλοι μηχανής.
Πολλές φορές όμως δεν έχουμε ιδανική επικάλυψη μεταξύ διαδοχικών εντολών, επειδή μία ή περισσότερες από αυτές χρησιμοποιεί το αποτέλεσμα κάποιας προηγούμενης που δεν το έχει ακόμα αποθηκεύσει στο φάκελο καταχωρητών. Αυτό το φαινόμενο το λέμε εξάρτηση από δεδομένα και είναι ένας από τους δύο σημαντικότερους παράγοντες που δεν επιτρέπουν ιδανική χρονική συμπεριφορά μερικά επικαλυπτόμενων εντολών. Έτσι στην περίπτωσή μας είδαμε ότι η δεύτερη εντολή χρησιμοποιεί το αποτέλεσμα της πρώτης από τον καταχωρητή $2, η τρίτη της δεύτερης από τον καταχωρητή $3, η πέμπτη της τρίτης και της τέταρτης από τους καταχωρητές $3 και $4 αντίστοιχα, ενώ η έκτη και η έβδομη χρησιμοποιούν το αποτέλεσμα της πέμπτης από τον καταχωρητή $4.
Αν υποθέσουμε ότι καμία εξαρτημένη εντολή δε μπορεί να προχωρήσει στη φάση ανάγνωσης τελούμενων πριν αυτά αποθηκευτούν από τις προηγούμενες εντολές από τις οποίες αυτή εξαρτάται, ο χρόνος ολοκλήρωσης των 7 εντολών θα είναι:

6 + 4 + 4 + 1 + 4 + 4 + 1 = 24

κύκλοι μηχανής, όπου ο χρόνος κάθε εντολής μετά την πρώτη προκύπτει από τον ένα επιπλέον κύκλο που επιβάλλει η επικάλυψη συν τους όποιους κύκλους αναμονής για κάποιο προηγούμενο αποτέλεσμα.
Θα μπορούσαμε να δούμε πώς προκύπτει ο παραπάνω χρόνος των 24 κύκλων, σχηματίζοντας το διάγραμμα χρονισμού της εκτέλεσης, δηλαδή το διάγραμμα που δείχνει τις εντολές που απασχολούν την κάθε φάση του κύκλου εντολής στο χρονικό διάστημα από την αρχή της πρώτης μέχρι το τέλος της τελευταίας από τις 7 εντολές:

| εντολή | cc1 | cc2 | cc3 | cc4 | cc5 | cc6 | cc7 | cc8 | cc9 | cc10 | cc11 | cc12 | cc13 | cc14 | cc15 | cc16 | cc17 | cc18 | cc19 | cc20 | cc21 | cc22 | cc23 | cc24 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | IF | ID | RF | EX | MEM | WB | | | | | | | | | | | | | | | | | | |
| 2 | | IF | ID | X | X | X | RF | EX | MEM | WB | | | | | | | | | | | | | | |
| 3 | | | IF | X | X | X | ID | X | X | X | RF | EX | MEM | WB | | | | | | | | | | |
| 4 | | | | | | | IF | X | X | X | ID | RF | EX | MEM | WB | | | | | | | | | |
| 5 | | | | | | | | | | | IF | ID | X | X | X | RF | EX | MEM | WB | | | | | |
| 6 | | | | | | | | | | | | IF | X | X | X | ID | X | X | X | RF | EX | MEM | WB | |
| 7 | | | | | | | | | | | | | | | | IF | X | X | X | ID | RF | EX | MEM | WB |

όπου IF = instruction fetch (ανάκληση εντολής), ID = instruction decode (αποκωδικοποίηση εντολής), RF = register fetch (ανάγνωση φακέλου καταχωρητών), EX = execute (εκτέλεση), MEM = memory access (προσπέλαση μνήμης δεδομένων) και WB = write back (αποθήκευση αποτελέσματος) είναι τα ονόματα των έξι επικαλυπτόμενων φάσεων του κύκλου εντολής. Με X δείχνουμε την απραξία (ή ανάσχεση ή πάγωμα) που προκύπτει από κάποια εξάρτηση. Παρατηρήστε ότι στη μερική επικάλυψη το πάγωμα μιας φάσης μεταφέρεται σε όλες τις προηγούμενες φάσεις, οι οποίες απασχολούνται από *επόμενες* εντολές.

Αν τέλος διαθέτουμε τη δυνατότητα εκτέλεσης εκτός σειράς, κάποιες εντολές που δεν εξαρτώνται από προηγούμενες μπορούν να προωθηθούν στη φάση εκτέλεσης και να εκτελεστούν, ενόσω κάποιες προηγούμενες περιμένουν τα αποτελέσματα άλλων εντολών.
Στην περίπτωσή μας, η μόνη εντολή που μπορεί να εκτελεστεί εκτός σειράς είναι η τέταρτη. Αν λοιπόν η εντολή αυτή εκτελεστεί νωρίτερα από τις προηγούμενες, για παράδειγμα από

την τρίτη που περιμένει τη δεύτερη, τότε κερδίζουμε έναν κύκλο μηχανής από τους 24 που υπολογίσαμε πιο πάνω. Το κέρδος θα ήταν μεγαλύτερο, εάν η πέμπτη εντολή εξαρτιόταν μόνο από την τέταρτη, και όχι από την τρίτη, ώστε να μπορεί να εκτελεστεί αμέσως μετά την τέταρτη, πιθανά πριν την ολοκλήρωση της τρίτης. Σε μια τέτοια περίπτωση θα κερδίζαμε έως και όλους τους κύκλους που τώρα η πέμπτη εντολή χρειάζεται να περιμένει την τρίτη.

**Άσκηση 2:**

*Α. Όταν ο χρόνος προσπέλασης μιας λέξης κύριας μνήμης σε ένα υπολογιστικό σύστημα 1GHz είναι 50ns, να βρεθεί το ελάχιστο ποσοστό χαμένου χρόνου λειτουργίας για ένα πρόγραμμα εφαρμογής, όταν η μέση συχνότητα εντολών προσπέλασης στη μνήμη είναι μία κάθε πέντε εντολές και ο ιδανικός ρυθμός ολοκλήρωσης διαδοχικών εντολών είναι μία ανά κύκλο μηχανής με τεχνική μερικής επικάλυψης. Υποθέτουμε ότι δεν έχουμε κρυφή μνήμη και καμία μορφή παραλληλίας στην προσπέλαση της κύριας μνήμης ή στην εκτέλεση των εντολών.*
*Β. Το ίδιο, εάν εισάγουμε κρυφή μνήμη με χρόνο προσπέλασης 2ns, με την υπόθεση ότι όλες οι προσπελάσεις γίνονται στην κρυφή μνήμη.*
*Γ. Το ίδιο, εάν μόνο το 90% των προσπελάσεων γίνεται στην κρυφή μνήμη (ρυθμός ευστοχίας 90%).*
*Δ. Η παραπάνω συχνότητα προσπέλασης στη μνήμη είναι ρεαλιστική για επεξεργαστές με καταχωρητές γενικού σκοπού. Σχολιάστε την περίπτωση επεξεργαστή που έχει μόνο έναν καταχωρητή (πχ συσσωρευτή), κι επομένως η συχνότητα προσπέλασης στη μνήμη θα είναι υψηλότερη.*

**Απάντηση:**

Σε έναν επεξεργαστή που λειτουργεί με την τεχνική της μερικής επικάλυψης, διαδοχικές εντολές μπορούν να ολοκληρώνονται ανά έναν κύκλο μηχανής. Η ιδανική αυτή συμπεριφορά διαταράσσεται όμως με διάφορους τρόπους. Εκτός από περιπτώσεις εξαρτήσεων στην εκτέλεση διαδοχικών εντολών, έχουμε και παράγοντες εκτός του επεξεργαστή που οδηγούν σε απώλεια κύκλων μηχανής στην εκτέλεση ενός προγράμματος. Για παράδειγμα, μια λειτουργία Ε/Ε απαιτεί πολλές φορές χιλιάδες κύκλους μηχανής, ενώ μια προσπέλαση κύριας μνήμης απαιτεί μερικές δεκάδες κύκλους για να ολοκληρωθεί. Σε όλες αυτές τις περιπτώσεις, και εφ' όσον δεν υπάρχει μηχανισμός παραλληλίας που να επιτρέπει άλλες λειτουργίες να εκτελούνται κατά τη διάρκεια ολοκλήρωσης της συγκεκριμένης λειτουργίας Ε/Ε ή προσπέλασης μνήμης, η εντολή του προγράμματος που την εκτελεί είναι αναγκασμένη να διακοπεί μέχρι την ολοκλήρωσή της. Ο χρόνος που μεσολαβεί κατά την αναμονή του προγράμματος μπορεί να θεωρηθεί χαμένος χρόνος για το συγκεκριμένο πρόγραμμα.

Με ρολόι 1GHz ο χρόνος του κύκλου μηχανής είναι 1ns. Έτσι:

Α. Στην περίπτωση που η προσπέλαση της κύριας μνήμης ολοκληρώνεται σε χρόνο 50ns χωρίς καμία δυνατότητα παραλληλίας, η επιβάρυνση του χρόνου εκτέλεσης του προγράμματος για την προσπέλαση αυτή είναι 50-1ns = 49ns για κάθε 5 εντολές, εφ’ όσον σε κάθε 5 εντολές έχουμε μία προσπέλαση της κύριας μνήμης, και υποθέτοντας ότι ο ένας κύκλος που αναλογεί με την τεχνική της μερικής επικάλυψης στην εντολή προσπέλασης μνήμης συμπεριλαμβάνεται στο χρόνο προσπέλασης. Έτσι, απαιτούνται συνολικά 54ns για ολοκλήρωση 5 διαδοχικών εντολών, όταν ιδανικά θα θέλαμε χρόνο ίσο με 5ns[4]. Επομένως, και δεδομένου ότι πιθανά θα έχουμε και άλλες απώλειες χρόνου, όπως για παράδειγμα για λειτουργίες Ε/Ε, ο ελάχιστος χαμένος χρόνος είναι 49 ανά 54ns, σε ποσοστό δηλαδή:

49/54 = 90.74%

του συνολικού χρόνου εκτέλεσης του προγράμματος.

[4] Εδώ αναφερόμαστε στο μέσο χρόνο ολοκλήρωσης διαδοχικών εντολών, υποθέτοντας ότι έχουμε μεγάλο συνολικό αριθμό εντολών. Έτσι, ο χρόνος εκτέλεσης μιας εντολής (6 κύκλοι) είναι αμελητέος σε σχέση με το συνολικό χρόνο εκτέλεσης του προγράμματος (πιθανά εκατομμύρια κύκλοι).

Β. Όταν έχουμε κρυφή μνήμη με χρόνο προσπέλασης 2ns και όλες τις προσπελάσεις να γίνονται σε αυτήν, η απώλεια χρόνου μειώνεται στο 1ns για κάθε 5 εντολές. Τώρα ο χρόνος ολοκλήρωσης 5 διαδοχικών εντολών θα είναι 6ns, και επομένως το ελάχιστο ποσοστό απώλειας χρόνου για το συγκεκριμένο πρόγραμμα εφαρμογής θα γίνεται:

1/6 = 16.67%

του χρόνου εκτέλεσης.
Είναι προφανές ότι η απώλεια χρόνου μειώνεται με τη μείωση του χρόνου προσπέλασης μνήμης, αλλά δε μπορεί να εξαλειφτεί όσο ο χρόνος προσπέλασης είναι μεγαλύτερος από το χρόνο κύκλου μηχανής του επεξεργαστή.

Γ. Στην περίπτωση αυτή, όταν δηλαδή το 90% των προσπελάσεων γίνεται στην κρυφή μνήμη και το υπόλοιπο 10% γίνεται στην κύρια μνήμη, ο χρόνος προσπέλασης μνήμης γίνεται 2ns για το 90% και 52ns για το 10% των περιπτώσεων προσπέλασης. Παρατηρήστε ότι εφ' όσον έχουμε ιεραρχία με κρυφή και κύρια μνήμη, ο χρόνος προσπέλασης της κύριας μνήμης αυξάνεται κατά το χρόνο προσπέλασης της κρυφής μνήμης, επειδή κάθε προσπέλαση στην κύρια μνήμη γίνεται με μεταφορά δεδομένων από την κύρια στην κρυφή μνήμη και στη συνέχεια με προσπέλαση στην τελευταία.
Ο μέσος χρόνος προσπέλασης μνήμης θα είναι:

90% * 2ns + 10% * 52ns = 7ns

κι επομένως ο χρόνος ολοκλήρωσης 5 διαδοχικών εντολών θα είναι τουλάχιστον 11ns, με ποσοστό απώλειας χρόνου:

6/11 = 54.55%

του χρόνου εκτέλεσης του προγράμματος.

Δ. Σε παλαιότερους επεξεργαστές που δε διέθεταν φάκελο καταχωρητών γενικού σκοπού, ο οποίος να μπορεί να αποθηκεύει ενδιάμεσα αποτελέσματα πράξεων της ΑΛΜ, κάθε εντολή ήταν αναγκασμένη να παίρνει ένα ή περισσότερα τελούμενα από τη μνήμη και πιθανά να αποθηκεύει το αποτέλεσμά της επίσης στη μνήμη. Βέβαια, οι επεξεργαστές αυτοί είχαν ρολόγια με συχνότητα πολύ μικρότερη από 1GHz, με χρόνους προσπέλασης μνήμης συγκρίσιμους με το χρόνο κύκλου μηχανής, έτσι ώστε το ποσοστό απώλειας χρόνου να μην είναι ιδιαίτερα υψηλό. Με ταχύτητα ρολογιού 1GHz όμως, το ποσοστό αυτό γίνεται εντυπωσιακά υψηλό, εάν η συχνότητα προσπελάσεων είναι για παράδειγμα μία προσπέλαση ανά εντολή.
Με μία προσπέλαση ανά εντολή, θα έχουμε ελάχιστα ποσοστά απώλειας χρόνου (Α) 49/50 = 98% για προσπέλαση στην κύρια μνήμη των 50ns, (Β) 1/2 = 50% για προσπέλαση στην κρυφή μνήμη των 2ns, και (Γ) 6/7 = 85.71% για ρυθμό επιτυχίας κρυφής μνήμης ίσο με 90%.