



Κεφάλαιο 4

**Σύνοψη των
περιπτώσεων της
διοχέτευσης (pipelining)**

Χρόνος κύκλου και CPI

- Στη σχεδίαση με διοχέτευση:
 - Ο χρόνος του κύκλου ρολογιού είναι ίσος με το πιο αργό από τα στάδια της διοχέτευσης:
 $T_{\text{pipeline}} = \max\{T_{\text{IF}}, T_{\text{ID}}, T_{\text{EX}}, T_{\text{MEM}}, T_{\text{WB}}\}$
 - Το ιδανικό CPI (δηλαδή χωρίς καθυστερήσεις δομής, δεδομένων, ελέγχου) ενός προγράμματος με I εντολές που εκτελείται σε μία της διοχέτευση k σταδίων ($k=5$ στον MIPS) είναι ίσο με:

$$CPI_{\text{ideal-pipeline}} = (I+k-1)/I \approx 1 \text{ (για μεγάλο } I)$$

- Σε αντιδιαστολή, στη σχεδίαση ενός κύκλου ρολογιού, ισχύουν:

- $T_{\text{single-cycle}} = T_{\text{IF}} + T_{\text{ID}} + T_{\text{EX}} + T_{\text{MEM}} + T_{\text{WB}}$

- $CPI_{\text{single-cycle}} = 1$



Καθυστερήσεις – κίνδυνοι

- Στη διοχέτευση μπορεί να εισαχθούν **καθυστερήσεις (stalls)** δηλαδή κύκλοι στους οποίους δεν μπορεί να ξεκινήσει η εκτέλεση μιας εντολής ή πρέπει να καθυστερήσει η εκτέλεση μιας εντολής που ήδη ξεκίνησε.
- Οι καθυστερήσεις οφείλονται σε **κινδύνους (hazards)**:
 - Δομής
 - Δεδομένων
 - Ελέγχου (Διακλάδωσης)



CPI με καθυστερήσεις

- Σε μια πραγματική διοχέτευση με κινδύνους το CPI είναι μεγαλύτερο από το ιδανικό και δίνεται από τη σχέση:
- $CPI_{\text{pipeline}} = CPI_{\text{ideal}}$
 - + Καθυστερήσεις δομής
 - + Καθυστερήσεις δεδομένων
 - + Καθυστερήσεις ελέγχου
- Το CPI_{ideal} αν δεν δοθεί κάτι διαφορετικό υποθέτουμε ότι είναι ίσο με 1



Κίνδυνοι δομής [1]

- Οι ενδεχόμενοι κίνδυνοι δομής στο pipeline του MIPS είναι δύο:
 - [1] Δεν υπάρχουν χωριστές μνήμες (εντολών και δεδομένων) αλλά μόνο μία ενιαία μνήμη – αυτό σημαίνει ότι δεν μπορεί να γίνει σε έναν κύκλο ρολογιού προσπέλαση για εντολή και προσπέλαση για δεδομένο. Στο σενάριο αυτό πρέπει να καθυστερήσει κατά 1 κύκλο η εντολή που ακολουθεί (για προσπέλαση εντολής) ώστε να ολοκληρωθεί η πρώτη που κάνει προσπέλαση δεδομένων
- **Η επιβάρυνση στο CPI είναι ένας επιπλέον κύκλος ρολογιού για κάθε εντολή load και store στο πρόγραμμα**



Κίνδυνοι δομής [1]

- Παράδειγμα: ένα πρόγραμμα έχει 20% εντολές load και 10% εντολές store. Εκτελείται σε μία CPU MIPS με διοχέτευση 5 σταδίων αλλά μόνο μία ενιαία μνήμη.
- Το CPI λόγω του κινδύνου δομής της ενιαίας μνήμης είναι:
- $CPI = CPI_{ideal} + (0.20 + 0.10) * 1 = 1 + 0.3 = 1.3$
- Διότι όλες οι εντολές έχουν CPI ίσο με 1 αλλά όσες «συναντούν» μία load ή store πρέπει να καθυστερήσουν το IF τους κατά 1 κύκλο.



Κίνδυνοι δομής [2]

- Ο δεύτερος ενδεχόμενος κίνδυνος δομής στο pipeline του MIPS είναι η αδυναμία εκτέλεσης δύο προσπελάσεων (μία εγγραφή και μία ανάγνωση) στους Καταχωρητές (αρχείο Καταχωρητών).
- Και σε αυτή την περίπτωση όταν δύο εντολές προσπαθούν στον ίδιο κύκλο να προσπελάσουν τους Καταχωρητές, η δεύτερη πρέπει να περιμένει (να «καθυστερήσει») ώστε να ολοκληρώσει η πρώτη την προσπέλαση των καταχωρητών.
- Ισχύουν τα ίδια για το CPI με την προηγούμενη περίπτωση.
- Στην πλειονότητα των περιπτώσεων θα υποθέτουμε ότι μπορεί να γίνει προσπέλαση των Καταχωρητών στον ίδιο κύκλο ρολογιού από δύο εντολές: η πρώτη γράφει σε έναν καταχωρητή και η δεύτερη διαβάζει.



Κίνδυνοι δεδομένων

- Όταν μία εντολή παράγει αποτέλεσμα (σε καταχωρητή ή μνήμη) και μια επόμενη το καταναλώνει.
- Δεν υφίσταται κίνδυνος δεδομένων σε εντολές που απέχουν περισσότερους από δύο κύκλους ρολογιού, διότι υποθέτουμε ότι όταν δύο εντολές «συναντιώνονται» στους Καταχωρητές (στο στάδιο WB ή πρώτη για να γράψει και στο στάδιο ID η δεύτερη για να διαβάσει) τότε όλα λειτουργούν σωστά: η πρώτη γράφει και η δεύτερη διαβάζει στον ίδιο κύκλο ρολογιού.
- Κίνδυνος δεδομένων μπορεί να υπάρχει μόνο μεταξύ μιας εντολής και της επόμενής της (1 κύκλος απόσταση) ή της μεθεπόμενής της (2 κύκλοι απόσταση)



Κίνδυνοι δεδομένων

- Όταν ένας επεξεργαστής με διοχέτευση δεν διαθέτει προώθηση (επόμενη τεχνική) για τους κινδύνους δεδομένων τότε πρέπει να «ανιχνεύει» τους κινδύνους και να προσθέτει καθυστερήσεις (stalls)
- Οι καθυστερήσεις πρέπει να είναι **2 κύκλοι ρολογιού εάν η εντολή που χρησιμοποιεί το αποτέλεσμα είναι η επόμενη**, και **1 κύκλος εάν είναι η μεθεπόμενη** (ήδη υπάρχει 1 κύκλος απόσταση σε αυτή την περίπτωση) – ώστε και στις δύο περιπτώσεις η εντολή που παράγει και η εντολή που καταναλώνει να συμπίσουν στον κύκλο WB της πρώτης και ID της δεύτερης.
- Παράδειγμα: εάν ένα πρόγραμμα έχει 10% εντολές που δίνουν το αποτέλεσμά τους στην επόμενη και 5% εντολές που δίνουν το αποτέλεσμά τους στην μεθεπόμενη τότε για το CPI ισχύει:
$$CPI = CPI_{ideal} + 0.10 * 2 + 0.05 * 1 = 1 + 0.20 + 0.05 = 1.25$$



Πρώθηση

- Η πρώθηση (forwarding) λύνει το πρόβλημα για τους κινδύνους δεδομένων που αφορούν καταχωρητές.
- Το αποτέλεσμα μεταφέρεται στην εντολή που το χρειάζεται ακριβώς στο στάδιο EX που κάνει την πράξη της και δεν υφίστανται καθόλου καθυστερήσεις. Άρα στο προηγούμενο παράδειγμα το CPI θα είναι 1 και δεν θα προστεθούν επιβαρύνσεις



Κίνδυνοι δεδομένων load/use

- Η προώθηση δεν μπορεί να λύσει πλήρως το πρόβλημα όμως όταν η εντολή που παράγει είναι μία load. Τότε και με την πλήρη προώθηση υλοποιημένη απαιτείται ένα κύκλος καθυστέρησης ανάμεσα στην load και την εντολή που χρησιμοποιεί το αποτέλεσμα.
- Παράδειγμα – αν ένα πρόγραμμα έχει 15% εντολές load που η επόμενη χρησιμοποιεί το αποτέλεσμά τους και εκτελείται σε μία διοχέτευση με πλήρη προώθηση τότε το CPI θα είναι:
- $$CPI = CPI_{ideal} + 0.15 * 1 = 1 + 0.15 = 1.15$$
- Η αναδιάταξη των εντολών μπορεί να βοηθήσει



Κίνδυνοι ελέγχου

- Εάν δεν κάνουμε πρόβλεψη διακλάδωσης τότε κάθε εντολή branch (beq, bne στον MIPS) πρέπει να «περιμένει» μέχρι να γνωρίζει εάν είναι αληθής ή ψευδής η συνθήκη της. Υπάρχουν δύο σενάρια για το σημείο που η συνθήκη γίνεται γνωστή (αυτό ονομάζεται «εκτέλεση» ή «επίλυση» της διακλάδωσης)
 - Στο στάδιο EX (κάνει τη σύγκριση η ALU) – απόσταση 2 κύκλοι ρολογιού από την έναρξη της branch
 - Στο στάδιο ID (η σύγκρισή γίνεται αμέσως μόλις διαβαστούν οι καταχωρητές – προσθέτει hardware για σύγκριση) – απόσταση 1 κύκλος ρολογιού από την έναρξη της branch



Κίνδυνοι ελέγχου

- Όταν μια διακλάδωση απλώς πρέπει να «περιμένει» την επίλυση λέμε ότι έχουμε stall-on-branch προσέγγιση, δηλαδή προστίθενται καθυστερήσεις (1 ή 2 κύκλοι) για κάθε διακλάδωση μέχρι να ξέρει η CPU από που θα προχωρήσει την εκτέλεση.
- Ακολουθούν παραδείγματα για το CPI



Κίνδυνοι ελέγχου

- Παράδειγμα – ένα πρόγραμμα με 15% διακλαδώσεις εκτελείται σε μια CPU MIPS με διοχέτευση 5 σταδίων και επίλυση των διακλαδώσεων στο στάδιο EX (επιβάρυνση 2 κύκλοι) και προσέγγιση stall-on-branch
- $CPI = CPI_{ideal} + 0.15 * 2 = 1 + 0.30 = 1.30$
- Το ίδιο πρόγραμμα εκτελείται στην ίδια CPU αλλά τώρα η επίλυση των διακλαδώσεων γίνεται στο στάδιο ID (επιβάρυνση 1 κύκλος)
- $CPI = CPI_{ideal} + 0.15 * 1 = 1 + 0.15 = 1.15$



Πρόβλεψη διακλάδωσης

- Είτε με δυναμικό (αλλάζει απόφαση) είτε με στατικό τρόπο (πάντα η ίδια απόφαση πρόβλεψης) η CPU «μαντεύει» προς τα που θα πάει μια διακλάδωση. Αν μαντέψει σωστά τότε δεν υπάρχει καμία καθυστέρηση. Αν μαντέψει λάθος τότε η επιβάρυνση των καθυστερήσεων (1 ή 2 κύκλοι ανάλογα με το που γίνεται η επίλυση της διακλάδωσης) «πληρώνεται» μόνο για τις εσφαλμένες προβλέψεις.
- Ακολουθεί παράδειγμα.



Πρόβλεψη διακλάδωσης

- Παράδειγμα – πρόγραμμα εκτελείται σε MIPS CPU με διοχέτευση 5 σταδίων, επίλυση των διακλαδώσεων στο στάδιο EX, 15% διακλαδώσεις από τις οποίες το 80% προβλέπονται σωστά ενώ το 20% προβλέπονται εσφαλμένα.
- $CPI = CPI_{ideal} + 0.15 * 0.20 * 2 = 1 + 0.06 = 1.06$
- Από το σύνολο των διακλαδώσεων (15%) μόνο το 20% (οι εσφαλμένα προβλεφθείσες) «πληρώνουν» τους 2 κύκλους επιβάρυνσης (είναι 2 επειδή η επίλυση γίνεται στο EX)



Καθυστερημένη διακλάδωση

- Κανόνας: η επόμενη εντολή κάθε εντολής διακλάδωσης εκτελείται υποχρεωτικά! Η CPU δεν βλέπει ποια είναι!
- Αν μπορούμε εκεί να μεταφέρουμε μια «χρήσιμη» εντολή που δεν εξαρτάται από τη διακλάδωση τότε δεν υπάρχει καθυστέρηση
- Αν δεν υπάρχει τέτοια εντολή τότε πρέπει να βάλουμε εμείς (δεν το κάνει η CPU) μία εντολή nop (που είναι η `sll $0,$0,0` στον MIPS)
- Εδώ το συνολικό CPI δεν θα επιβαρυνθεί αλλά αφού οι επιπλέον υποχρεωτικές nop είναι «άχρηστες» εντολές ουσιαστικά επιβαρύνουν την εκτέλεση με επιπλέον CPI.
- Παράδειγμα: ένα πρόγραμμα εκτελείται σε επεξεργαστή MIPS με διοχέτευση και καθυστερημένες διακλαδώσεις. Το πρόγραμμα περιέχει 20% διακλαδώσεις. Για το 30% των διακλαδώσεων πρέπει να εισαχθούν εντολές nop για να εκτελεστεί σωστά το πρόγραμμα (για τις υπόλοιπες διακλαδώσεις εισάγεται χρήσιμη εντολή μετά τη διακλάδωση).
- $$CPI = CPI_{ideal} + 0.20 * 0.30 * 1 = 1 + 0.06 = 1.06$$

