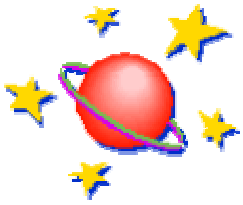

Ενότητα 7η : *Η τεχνική της πολλαπλής διευθέτησης*

Σκοπός

Ο σκοπός της ενότητας αυτής είναι να παρουσιάσει τους επεξεργαστές πολλαπλής διευθέτησης. Θα αναλύσουμε τους υπερβαθμωτούς επεξεργαστές που είναι είτε στατικά προγραμματισμένοι από τον μεταγλωττιστή, είτε δυναμικά χρησιμοποιώντας τις τεχνικές του πίνακα αποτελεσμάτων και του Tomasulo. Τέλος, θα αναφερθούμε αναλυτικά στους επεξεργαστές με πολύ μεγάλη λέξη εντολής (VLIW), οι οποίοι είναι στατικά προγραμματισμένοι από τον μεταγλωττιστή.

Προσδοκώμενα Αποτελέσματα

Όταν θα έχετε μελετήσει την ενότητα, θα είστε σε θέση:

<u>να περιγράφετε τους υπερβαθμωτούς επεξεργαστές και συγκεκριμένα τους υπερβαθμωτούς DLX επεξεργαστές,</u>

<u>να εξηγείτε τους τρόπους με τους οποίους επιτυγχάνεται η πολλαπλή διευθέτηση των εντολών,</u>

<u>να προσδιορίζετε τους περιορισμούς που υπάρχουν στην υλοποίηση των επεξεργαστών πολλαπλής διευθέτησης,</u>

<u>να περιγράφετε τους επεξεργαστές με πολύ μεγάλη λέξη εντολής (VLIW).</u>

πολλαπλή διευθέτηση εντολών, επεξεργαστής πολλαπλής διευθέτησης, υπερβαθμωτός επεξεργαστής, επεξεργαστής με πολύ μεγάλη λέξη εντολής (VLIW), μη συζευγμένη αρχιτεκτονική

Γενικά για τους VLIW και υπερβαθμωτούς επεξεργαστές

Για να βελτιώσουμε την απόδοση της παραλληλίας σε επίπεδο εντολής, πρέπει να μειώσουμε το CPI ώστε να γίνει μικρότερο της μονάδας. Όμως, αν διευθετούμε μόνο μία εντολή σε κάθε κύκλο ρολογιού το CPI δεν μπορεί να μειωθεί.

Οι επεξεργαστές πολλαπλής διευθέτησης επιτρέπουν σε πολλαπλές εντολές να διευθετούνται σε ένα κύκλο ρολογιού. Η μεγαλύτερη πρόκληση για όλους τους επεξεργαστές πολλαπλής διευθέτησης είναι να προσπαθήσουν να εκμεταλλευτούν όσο γίνεται την παραλληλία σε επίπεδο εντολής.

Υπάρχουν δύο κατηγορίες επεξεργαστών πολλαπλής διευθέτησης**:**

1. Υπερβαθμωτοί επεξεργαστές.

2. Επεξεργαστές με πολύ μεγάλη λέξη εντολής (VLIW).

Οι **υπερβαθμωτοί** επεξεργαστές διευθετούν μεταβαλλόμενο αριθμό εντολών σε κάθε κύκλο ρολογιού. Αυτοί είναι είτε στατικά προγραμματισμένοι από τον μεταγλωττιστή, είτε δυναμικά χρησιμοποιώντας τις τεχνικές του πίνακα αποτελεσμάτων και του Tomasulo.

Οι **VLIW** επεξεργαστές διευθετούν ένα καθορισμένο αριθμό εντολών, σε κάθε κύκλο ρολογιού, διαμορφωμένο είτε σαν μια μεγάλη εντολή, είτε σαν ένα καθορισμένο πακέτο εντολών. Αυτοί είναι στατικά προγραμματισμένοι από τον μεταγλωττιστή.

Σε όλα τα παραδείγματα που θα ακολουθήσουν υποθέτουμε ότι έχουμε τις καθυστερήσεις του σχήματος 2.1.1 και το τμήμα κώδικα που προσθέτει μία βαθμωτή τιμή σε έναν πίνακα (ενότητα 1)**:**

```
loop: LD     F0, 0 (R1)  ;   F0 = στοιχείο πίνακα
      ADDD   F4, F0, F2  ;   πρόσθεση με το βαθμωτό μέγεθος του F2
      SD     0 (R1), F4  ;   αποθήκευση του αποτελέσματος
      SUBI   R1, R1, #8  ;   μείωση δείκτη 8 bytes (για κάθε DW)
      BNEZ   R1, loop    ;   διακλάδωση R1!= μηδέν
```

Υπερβαθμωτή έκδοση του DLX επεξεργαστή

Ένας τυπικός υπερβαθμωτός επεξεργαστής διευθετεί από μία έως οκτώ εντολές σε κάθε κύκλο ρολογιού. Οι εντολές αυτές θα πρέπει να είναι ανεξάρτητες και να μην υπάρχουν περισσότερες από μία εντολές αναφοράς στη μνήμη σε κάθε κύκλο ρολογιού. Αν δεν ισχύει κάποιο από αυτά, τότε διευθετούνται οι εντολές που βρίσκονται πριν από την εντολή που δεν πληροί τις προϋποθέσεις. Σύμφωνα λοιπόν με τον αριθμό των εντολών που διευθετούνται, προκύπτει και η μεταβολή του ρυθμού διευθέτησης.

Ο υπερβαθμωτός επεξεργαστής έχει δυνατότητα δυναμικής διευθέτησης των εντολών.

Παράδειγμα

Έστω ότι διευθετούμε δύο εντολές σε κάθε κύκλο ρολογιού. Η μία μπορεί να είναι εντολή φόρτωσης, αποθήκευσης, διακλάδωσης με συνθήκη ή μια ακέραια αριθμητική ή λογική εντολή και η άλλη οποιαδήποτε εντολή κινητής υποδιαστολής.

Απάντηση:

Η δεύτερη εντολή διευθετείται μόνο αν μπορεί να διευθετηθεί η πρώτη εντολή. Στο παρακάτω πίνακα φαίνονται οι εντολές όταν μπαίνουν στη σωλήνωση σε ζεύγη.

Τύπος Εντολής	**Στάδια Σωλήνωσης**							
Ακέραια εντολή	IF	ID	EX	MEM	WB			
FP εντολή	IF	ID	EX	MEM	WB			
Ακέραια εντολή		IF	ID	EX	MEM	WB		
FP εντολή		IF	ID	EX	MEM	WB		
Ακέραια εντολή			IF	ID	EX	MEM	WB	
FP εντολή			IF	ID	EX	MEM	WB	
Ακέραια εντολή				IF	ID	EX	MEM	WB
FP εντολή				IF	ID	EX	MEM	WB

Οι εντολές ακεραίων και κινητής υποδιαστολής διευθετούνται ταυτόχρονα και η καθεμιά εκτελείται με το δικό της ρυθμό μέσα στη σωλήνωση.

Η σωλήνωση αυτή ενισχύει σημαντικά το ρυθμό διευθέτησης των εντολών κινητής υποδιαστολής. Επομένως, χρειαζόμαστε είτε σωληνωμένες μονάδες κινητής υποδιαστολής, είτε πολλαπλές ανεξάρτητες μονάδες.

ΔΡΑΣΤΗΡΙΟΤΗΤΑ 1

Μπορείτε να προσδιορίσετε το λόγο για τον οποίο η παράλληλη διευθέτηση μιας εντολής ακεραίου και μιας εντολής κινητής υποδιαστολής ελαχιστοποιεί την ανάγκη για επιπλέον υλικό; Να αιτιολογήσετε την απάντησή σας.

ΑΠΑΝΤΗΣΗ ΔΡΑΣΤΗΡΙΟΤΗΤΑΣ 1

Διευθετώντας δύο εντολές παράλληλα, μία ακέραια και μία κινητής υποδιαστολής, η ανάγκη για επιπλέον υλικό ελαχιστοποιείται. Αυτή είναι και η ιδανική περίπτωση. Αν δεν ισχύει αυτό, τότε οι εντολές διευθετούνται διαδοχικά. Στην περίπτωση αυτή, για την εκτέλεση των εντολών ακεραίων και κινητής υποδιαστολής χρησιμοποιούνται διαφορετικές ομάδες καταχωρητών και διαφορετικές λειτουργικές μονάδες στις αρχιτεκτονικές φόρτωσης αποθήκευσης.

ΔΡΑΣΤΗΡΙΟΤΗΤΑ 2

Έστω ότι έχουμε δύο εντολές κινητής υποδιαστολής στη σειρά, οι οποίες μπορεί να είναι εντολές φόρτωσης, εντολές αποθήκευσης ή εντολές διακλάδωσης με συνθήκη.

- **Ποιες θεωρείται πως είναι οι δυσκολίες που προκύπτουν από την διευθέτηση των εντολών αυτών;**
- **Να προτείνετε τρόπο αντιμετώπισης των δυσκολιών που προσδιορίσατε.**

ΑΠΑΝΤΗΣΗ ΔΡΑΣΤΗΡΙΟΤΗΤΑΣ 2

- **Αν έχω δύο εντολές κινητής υποδιαστολής στη σειρά, εμφανίζονται οι εξής δυσκολίες:**
  1. **Δημιουργείται σύγκρουση για τις θύρες του αρχείου καταχωρητών κινητής υποδιαστολής.**
  2. **Μπορεί να δημιουργηθεί ένας νέος RAW κίνδυνος όταν η δεύτερη εντολή κινητής υποδιαστολής που διευθετείται στον ίδιο κύκλο ρολογιού εξαρτάται από την πρώτη εντολή του ζευγαριού.**
- **Η πρώτη δυσκολία, που αποτελεί και κατασκευαστικό κίνδυνο, μπορεί να αντιμετωπιστεί αν οι κινητής υποδιαστολής φορτώσεις και αποθηκεύσεις διευθετηθούν μόνες τους. Αυτό είναι εύκολο να εφαρμοστεί, αλλά έχει ουσιώδη μειονεκτήματα στην απόδοση. Εναλλακτικά, αυτός ο κίνδυνος θα μπορούσε να εξαλειφθεί παρέχοντας δύο επιπλέον θύρες, μία για ανάγνωση και μία για εγγραφή στο αρχείο καταχωρητών κινητής υποδιαστολής. Στη δεύτερη περίπτωση ο κίνδυνος θα μπορούσε να ανιχνευτεί και να μην διευθετηθεί η εντολή κινητής υποδιαστολής.**

Όταν το ζεύγος εντολών που ανακαλείται αποτελείται από μία εντολή φόρτωσης κινητής υποδιαστολής και από μία οποιαδήποτε εντολή κινητής υποδιαστολής που εξαρτάται από την προηγούμενη, τότε πρέπει να ανιχνεύσουμε τον κίνδυνο και να μην διευθετήσουμε την εντολή κινητής υποδιαστολής.

Υπάρχει άλλη μια δυσκολία που περιορίζει την αποτελεσματικότητα της υπερβαθμωτής σωλήνωσης:

- Το αποτέλεσμα μιας εντολής φόρτωσης δεν μπορεί να χρησιμοποιηθεί στον ίδιο κύκλο ρολογιού ή στον επόμενο.

Αυτό σημαίνει ότι οι επόμενες τρεις εντολές δεν μπορούν να χρησιμοποιήσουν το αποτέλεσμα της εντολής φόρτωσης χωρίς να καθυστερήσουν. Το ίδιο συμβαίνει και όταν η πρώτη εντολή είναι εντολή διακλάδωσης.

Παράδειγμα

Ας δούμε πώς προγραμματίζεται στην υπερβαθμωτή σωλήνωση ο παρακάτω βρόχος:

```
loop: LD      F0, 0 (R1)  ;   F0 = στοιχείο πίνακα
      ADDD    F4, F0, F2  ;   πρόσθεση με το βαθμωτό μέγεθος του F2
      SD      0 (R1), F4  ;   αποθήκευση του αποτελέσματος
      SUBI    R1, R1, #8  ;   μείωση δείκτη 8 bytes (για κάθε DW)
      BNEZ    R1, loop    ;   διακλάδωση R1!= μηδέν
```

Απάντηση:

Για να προγραμματίσουμε το βρόχο χωρίς καθυστερήσεις, χρειάζεται να αναπτύξουμε το βρόχο ώστε να φτιάξουμε πέντε αντίγραφα του κυρίου μέρους του. Μετά την ανάπτυξη, ο βρόχος θα περιέχει πέντε αντίγραφα των εντολών LD, ADDD και SD και ένα αντίγραφο των εντολών SUBI και BNEZ. Ο ανεπτυγμένος και προγραμματισμένος βρόχος φαίνεται στον παρακάτω πίνακα.

Ακέραια εντολή	FP εντολή	Κύκλοι ρολογιού
loop: LD F0, 0 (R1)		1
LD F6, -8 (R1)		2
LD F10, -16 (R1)	ADDD F4, F0, F2	3
LD F14, -24 (R1)	ADDD F8, F6, F2	4
LD F18, -32 (R1)	ADDD F12, F10, F2	5
SD 0 (R1), F4	ADDD F16, F14, F2	6
SD -8 (R1), F8	ADDD F20, F18, F2	7
SD -16 (R1), F12		8
SUBI R1, R1, #40		9
SD -24 (R1), F16		10
BNEZ R1, loop		11
SD -8 (R1), F20		12

Κάθε μία από τις πέντε εντολές ADDD διευθετείται μαζί με μια εντολή LD ή με μία εντολή SD, αλλά στη δεύτερη στήλη δεν υπάρχουν αρκετές εντολές κινητής υποδιαστολής ώστε να κρατήσουν γεμάτη τη σωλήνωση. Ο ανεπτυγμένος και προγραμματισμένος βρόχος στην υπερβαθμωτή σωλήνωση θα εκτελεστεί σε 12 κύκλους ρολογιού για κάθε επανάληψή του ή σε 2.4 κύκλους ρολογιού για κάθε ένα από τα πέντε αντίγραφα του βρόχου, σε αντίθεση με τον προγραμματισμένο και ανεπτυγμένο βρόχο στη συνηθισμένη DLX σωλήνωση όπου εκτελείται σε 16 κύκλους ρολογιού για κάθε επανάληψή του ή σε 3.2 κύκλους ρολογιού για κάθε ένα από τα πέντε αντίγραφα του βρόχου. Η απόδοση του επεξεργαστή περιορίζεται από την ισορροπία μεταξύ των υπολογισμών που αναφέρονται σε πράξεις ακεραίων ή κινητής υποδιαστολής.

Μέχρι τώρα έχουμε αναφερθεί σε υπερβαθμωτούς επεξεργαστές στατικά προγραμματισμένους από τον μεταγλωττιστή, ο οποίος είναι υπεύθυνος να βρει ανεξάρτητες εντολές ώστε να τις διευθετήσει.

Πολλαπλή διευθέτηση εντολών με δυναμικό προγραμματισμό

Η πολλαπλή διευθέτηση εντολών μπορεί επίσης να εφαρμοστεί και στους δυναμικά προγραμματισμένους υπερβαθμωτούς επεξεργαστές. Ας υποθέσουμε ότι θέλουμε να επεκτείνουμε τον αλγόριθμο του Tomasulo, ώστε να υποστηρίξουμε τη διευθέτηση δύο εντολών ανά κύκλο ρολογιού, μιας εντολής ακεραίων και μιας εντολής κινητής υποδιαστολής.
Μπορούμε να διευθετήσουμε ταυτόχρονα μια εντολή κινητής υποδιαστολής και μια εντολή ακεραίων στους αντίστοιχους σταθμούς κράτησής τους, όσο οι δύο διευθετημένες εντολές δεν έχουν πρόσβαση στην ίδια ομάδα καταχωρητών.

ΔΡΑΣΤΗΡΙΟΤΗΤΑ 3

Θυμάστε τι ποια είναι η χρησιμότητα των σταθμών κράτησης στην υλοποίηση της τεχνικής Tomasulo; Για περισσότερες λεπτομέρειες καλό θα ήταν να ανατρέξετε στην 5η ενότητα αυτού του κεφαλαίου: «Η μέθοδος Tomasulo».

ΑΠΑΝΤΗΣΗ ΔΡΑΣΤΗΡΙΟΤΗΤΑΣ 3

Στον αλγόριθμο Tomasulo η μετονομασία των καταχωρητών πραγματοποιείται με την βοήθεια των σταθμών κράτησης. Οι σταθμοί κράτησης ανακαλούν και καταχωρούν προσωρινά έναν τελεστέο μιας εντολής, μόλις βέβαια γίνει διαθέσιμος. Με αυτό τον τρόπο δε χρειάζεται να επαναληφθεί η ανάγνωση ενός συγκεκριμένου τελεστέου από έναν καταχωρητή, κατά τη διάρκεια εκτέλεσης του προγράμματος.

Η προσέγγιση της πολλαπλής διευθέτησης εντολών με δυναμικό προγραμματισμό, εμποδίζει τη διευθέτηση δύο εντολών που έχουν εξάρτηση στον ίδιο κύκλο ρολογιού, όπως είναι η εντολή φόρτωσης και η εντολή πρόσθεσης. Ωστόσο, θα θέλαμε να τις διευθετήσουμε στους σταθμούς κράτησης, όπου θα περιμένουν να εκτελεστούν σε μία λειτουργική μονάδα.

Υπάρχουν δύο προσεγγίσεις ώστε να επιτύχουμε διπλή διευθέτηση:

- Η πρώτη είναι να σωληνώσουμε το τμήμα διευθέτησης εντολών ώστε να τρέχει με το διπλάσιο του βασικού ρυθμού ρολογιού. Αυτό επιτρέπει την ενημέρωση των πινάκων πριν επεξεργαστούμε την επόμενη εντολή και έτσι οι δύο εντολές μπορούν να αρχίσουν την εκτέλεσή τους στον ίδιο κύκλο ρολογιού.

- Η δεύτερη υποθέτει, ότι σύμφωνα με τους περιορισμούς που έχουν ληφθεί, μόνο οι εντολές φορτώσεων και μετακινήσεων, από τους καταχωρητές ακεραίων (GP) στους καταχωρητές κινητής υποδιαστολής (FP), θα δημιουργήσουν εξαρτήσεις ανάμεσα σε εντολές που θα διευθετήσουμε μαζί.

Η ανάγκη πινάκων που καταγράφουν τα αποτελέσματα των σταθμών κράτησης εξαλείφεται, χρησιμοποιώντας ουρές για το αποτέλεσμα μιας εντολής φόρτωσης ή μετακίνησης. Επίσης χρησιμοποιούνται και για μια εντολή αποθήκευσης, ώστε να διευθετηθεί νωρίτερα και να περιμένει για τους τελεστέους της.
Αφού ο δυναμικός προγραμματισμός είναι περισσότερο αποτελεσματικός στις μετακινήσεις δεδομένων, μπορούμε να χρησιμοποιήσουμε στατικό προγραμματισμό για να εξαλείψουμε τελείως τους σταθμούς κράτησης και να βασιστούμε μόνο στις ουρές.

Αυτός ο τρόπος οργάνωσης του επεξεργαστή, όπου οι μονάδες φόρτωσης-αποθήκευσης έχουν ουρές που επιτρέπουν σύνδεση με άλλες λειτουργικές μονάδες, ονομάζεται μη συζευγμένη αρχιτεκτονική.

Ενας επεξεργαστής που προγραμματίζει δυναμικά τις εντολές φόρτωσης και αποθήκευσης, μπορεί να προκαλέσει αναδιάταξη αυτών των εντολών. Η αναδιάταξη μπορεί να οδηγήσει στην παραβίαση μιας εξάρτησης δεδομένων και να προκύψει κίνδυνος. Για να ανιχνεύσουμε τέτοιους κινδύνους ελέγχουμε δυναμικά αν η διεύθυνση μνήμης των πηγαίων τελεστέων μιας εντολής φόρτωσης είναι η ίδια με τη διεύθυνση προορισμού μιας μη ολοκληρωμένης εντολής αποθήκευσης. Αν είναι ίδια, τότε καθυστερούμε την εντολή φόρτωσης μέχρι να ολοκληρωθεί η εντολή αποθήκευσης. Αφού υπολογιστεί η διεύθυνση προορισμού της εντολής αποθήκευσης, τοποθετείται στον προσωρινό καταχωρητή αποθήκευσης. Υπάρχει επίσης η πιθανότητα WAW και WAR κινδύνων μέσω μνήμης.

Παράδειγμα

Ας υποθέσουμε ότι διευθετούμε δύο εντολές που είναι εξαρτώμενες αλλά χρησιμοποιούν διαφορετικές λειτουργικές μονάδες. Θεωρούμε την εκτέλεση του αρχικού βρόχου σε DLX σωλήνωση που έχει όμως επεκταθεί με τον αλγόριθμο Tomasulo και με την πολλαπλή διευθέτηση. Επίσης, θεωρούμε ότι η εντολή ADDD και η εντολή LD διευθετούνται μαζί σε κάθε κύκλο ρολογιού, ακόμα και αν υπάρχει μεταξύ τους εξάρτηση, με τη προϋπόθεση ότι η εντολή LD είναι πρώτη. Υποθέτουμε μία ακέραια λειτουργική μονάδα και μία ξεχωριστή λειτουργική μονάδα κινητής υποδιαστολής για κάθε λειτουργία. Ο αριθμός των κύκλων καθυστέρησης για κάθε εντολή είναι ο ίδιος. Υποθέτουμε ότι τα αποτελέσματα της διευθέτησης και της εγγραφής χρειάζονται ένα κύκλο ρολογιού το καθένα και ότι υπάρχει υλικό δυναμικής πρόβλεψης διακλάδωσης. Δημιουργήστε ένα πίνακα που να δείχνει πότε διευθετείται κάθε εντολή, πότε αρχίζει να εκτελείται και να γράφει το αποτέλεσμά της για τις δύο πρώτες επαναλήψεις του βρόχου. Δίνεται ο αρχικός βρόχος:

```
loop: LD      F0, 0 (R1)  ;   F0 = στοιχείο πίνακα
      ADDD    F4, F0, F2  ;   πρόσθεση με το βαθμωτό μέγεθος του
                              F2
      SD      0 (R1), F4  ;   αποθήκευση του αποτελέσματος
      SUBI    R1, R1, #8  ;   μείωση δείκτη 8 bytes (για κάθε DW)
      BNEZ    R1, loop    ;   διακλάδωση R1!= μηδέν
```

Απάντηση:

Ο βρόχος θα αναπτυχθεί δυναμικά και όπου αυτό είναι δυνατό οι εντολές θα διευθετούνται σε ζευγάρια. Ο βρόχος τρέχει σε 4 κύκλους ρολογιού ανά αποτέλεσμα, υποθέτοντας ότι δεν υπάρχουν καθυστερήσεις στην έξοδο του βρόχου. Τα αποτελέσματα φαίνονται στον παρακάτω πίνακα.

Αριθμός Επαναλήψεων	Εντολές	Διευθετήσεις σε κύκλους ρολογιού	Εκτελέσεις σε κύκλους ρολογιού	Πρόσβαση μνήμης σε κύκλους ρολογιού	Εγγραφές αποτελέσματος σε κύκλους ρολογιού
1	LD F0, 0(R1)	1	2	3	3
1	ADDD F4, F0, F2	1	4		7
1	SD 0(R1), F4	2	3	7	
1	SUBI R1, R1, #8	3	4		5
1	BNEZ R1, loop	4	5		
2	LD F0, 0(R1)	5	6	8	8
2	ADDD F4, F0, F2	5	8		11
2	SD 0(R1), F4	6	7	12	
2	SUBI R1, R1, #8	7	8		9
2	BNEZ R1, loop	8	9		

Το στάδιο εγγραφής αποτελέσματος δεν έχει τιμή για τις εντολές αποθήκευσης και διακλάδωσης, αφού δεν γράφουν σε καταχωρητές. Ο βρόχος τρέχει σε 7/2=3.5 κύκλους ρολογιού ανά αποτέλεσμα για 2 επαναλήψεις.
Υπάρχει η υπόθεση ότι κάθε αποτέλεσμα γράφεται στη CDB (common data bus) δηλαδή στη κοινή αρτηρία δεδομένων στο τέλος του κύκλου ρολογιού που είναι διαθέσιμο. Οπότε πρέπει να γίνουν οι αναγκαίες αλλαγές στον πίνακα. Ακόμα, για να εκτελεστεί η εντολή ADDD πρέπει να έχει τελειώσει η εγγραφή του αποτελέσματος (κοινή αρτηρία δεδομένων και μετά καταχωρητές ή σταθμοί κράτησης) της εντολής φόρτωσης ώστε να χρησιμοποιηθεί το αποτέλεσμά της.
Η εκτέλεση της εντολής αποθήκευσης αποτελέσματος πρέπει να εκτελεστεί μετά την εγγραφή του αποτελέσματος στην αρτηρία δεδομένων, οπότε η πρώτη εντολή SD θα έχει πρόσβαση στη μνήμη για να γράψει στον $7^{ο}$ κύκλο του ρολογιού. Αυτό παρασύρει και την δεύτερη εντολή LD που πρέπει να εκτελεστεί αργότερα, οπότε και η δεύτερη ADDD πρέπει να εκτελεστεί ένα κύκλο αργότερα. Τέλος, η δεύτερη εντολή SD πρέπει να εκτελεστεί μετά το τέλος της ADDD για να αποθηκεύσει τη σωστή τιμή του F4 αφού θα έχει γίνει η πρόσθεση.
Ο αριθμός των διπλών διευθετήσεων είναι μικρός επειδή υπάρχει μία εντολή κινητής υποδιαστολής ανά επανάληψη. Αν ο επεξεργαστής ήταν ευρύτερος θα διευθετούσε περισσότερες εντολές ακεραίων ανά κύκλο ρολογιού.

Οι VLIW επεξεργαστές

Με έναν VLIW επεξεργαστή μπορούμε να μειώσουμε την ποσότητα του υλικού που χρειάζεται για να πραγματοποιήσουμε έναν επεξεργαστή πολλαπλής διευθέτησης.

Η LIW ή VLIW αρχιτεκτονική χρησιμοποιεί πολλαπλές, ανεξάρτητες λειτουργικές μονάδες. Με την VLIW αρχιτεκτονική οι πολλαπλές εντολές ομαδοποιούνται σε μια πολύ μεγάλη εντολή που καλείται όνομα. Η ευθύνη για την επιλογή των εντολών που θα διευθετηθούν ταυτόχρονα, ανήκει στον μεταγλωττιστή.

Μία VLIW εντολή μπορεί να περιέχει δύο εντολές ακεραίων, δύο εντολές κινητής υποδιαστολής, δύο αναφορές στη μνήμη και μία εντολή διακλάδωσης. Μία εντολή μπορεί να έχει ένα σύνολο πεδίων για κάθε λειτουργική μονάδα - ίσως από 16 ως 24 bits ανά μονάδα – παράγοντας μία εντολή με μήκος από 112 ως 168 bits. Για να είναι απασχολημένες οι λειτουργικές μονάδες, πρέπει να υπάρχει αρκετή παραλληλία σε μια ακολουθία κώδικα χωρίς διακλαδώσεις, ώστε να γεμίζουν οι διαθέσιμες θύρες λειτουργίας. Αυτή η παραλληλία αποκαλύπτεται με την ανάπτυξη και τον προγραμματισμό των βρόχων χρησιμοποιώντας τεχνικές καθολικού προγραμματισμού. Ας υποθέσουμε ότι διαθέτουμε μία τεχνική που να παράγει μεγάλες ακολουθίες κώδικα χωρίς διακλαδώσεις, χρήσιμες στην αύξηση των VLIW εντολών.

Παράδειγμα

Θεωρούμε ότι έχουμε έναν VLIW επεξεργαστή που μπορεί να διευθετήσει δύο αναφορές στη μνήμη, δύο εντολές κινητής υποδιαστολής και μια εντολή ακεραίων ή διακλάδωσης σε κάθε κύκλο ρολογιού. Δείξτε μια εκδοχή ανάπτυξης του βρόχου για έναν τέτοιο επεξεργαστή. Αναπτύξτε το βρόχο όσες φορές είναι απαραίτητο για να εξαλειφθούν οι όποιες καθυστερήσεις. Αγνοείστε τη θύρα καθυστέρησης διακλάδωσης. Δίνεται ο βρόχος:

```
loop: LD      F0, 0 (R1)  ;   F0 = στοιχείο πίνακα
      ADDD    F4, F0, F2  ;    πρόσθεση με το βαθμωτό μέγεθος του
                               F2
      SD      0 (R1), F4  ;   αποθήκευση του αποτελέσματος
      SUBI    R1, R1, #8  ;   μείωση δείκτη 8 bytes (για κάθε DW)
      BNEZ    R1, loop    ;   διακλάδωση R1!= μηδέν
```

Απάντηση:

Ο βρόχος αναπτύχθηκε έτσι ώστε να δημιουργηθούν εφτά αντίγραφά του, γεγονός που εξαλείφει όλες τις καθυστερήσεις. Τα εφτά αποτελέσματα δίνονται σε 9 κύκλους ρολογιού ή σε 1.29 κύκλους ρολογιού ανά αποτέλεσμα. Οι VLIW εντολές της ανεπτυγμένης ακολουθίας φαίνονται στον παρακάτω πίνακα.

Αναφορά Μνήμης 1	**Αναφορά Μνήμης 2**	**Εντολή FP 1**	**Εντολή FP 2**	**Εντολή ακεραίων /διακλάδωσης με συνθήκη**
LD F0, 0(R1)	LD F6, -8(R1)			
LD F10, -16(R1)	LD F14, -24(R1)			
LD F18, -32(R1)	LD F22, -40(R1)	ADDD F4, F0, F2	ADDD F8, F6, F2	
LD F26, -48(R1)		ADDD F12, F10, F2	ADDD F16, F14, F2	
		ADDD F20, F18, F2	ADDD F24, F22, F2	
SD 0(R1), F4	SD -8(R1), F8	ADDD F28, F26, F2		
SD -16(R1), F12	SD -24(R1), F16			
SD -32(R1), F20	SD -40(R1), F24			SUBI R1, R1, #56
SD 8(R1), F28				BNEZ R1, loop

Ο ρυθμός διευθέτησης είναι 23 εντολές σε εννέα κύκλους ρολογιού ή 2.5 εντολές ανά κύκλο ρολογιού. Για να επιτύχουμε τέτοιο ρυθμό διευθέτησης, απαιτείται μεγαλύτερος αριθμός καταχωρητών από ότι στον απλό DLX επεξεργαστή. Η αποδοτικότητα, δηλαδή το ποσοστό των διαθέσιμων θυρών που περιείχαν μία λειτουργία, είναι περίπου 60%. Η παραπάνω ακολουθία κώδικα του VLIW επεξεργαστή απαιτεί τουλάχιστον οκτώ καταχωρητές κινητής υποδιαστολής, ενώ ο βασικός DLX επεξεργαστή απαιτεί από δύο ως πέντε και ο υπερβαθμωτός έξι καταχωρητές.

Επειδή ένας VLIW είναι στατικά προγραμματισμένος και λειτουργεί με κλείδωση βήματος, μία καθυστέρηση σε οποιαδήποτε λειτουργική μονάδα καθυστερεί ολόκληρο τον επεξεργαστή, αφού όλες οι λειτουργικές μονάδες είναι συγχρονισμένες.

Περιορισμοί στους επεξεργαστές πολλαπλής διευθέτησης

Αν μπορούμε να διευθετήσουμε πέντε εντολές ανά κύκλο ρολογιού, γιατί να μην μπορούμε και 50. Η δυσκολία αύξησης του ρυθμού διευθέτησης προέρχεται από τρεις περιοχές:

1. Έμφυτοι περιορισμοί της παραλληλίας σε επίπεδο εντολών στα προγράμματα.
2. Δυσκολίες στην κατασκευή του υλικού.
3. Ειδικοί περιορισμοί για υπερβαθμωτή ή VLIW εφαρμογή.

Σε έναν στατικά προγραμματισμένο επεξεργαστή μπορεί να μην υπάρχουν αρκετές λειτουργίες για να γεμίσουν τις διαθέσιμες θύρες διευθέτησης εντολών, εκτός αν οι βρόχοι είναι ανεπτυγμένοι πολλές φορές. Ίσως να φαίνεται ότι πέντε εντολές, που μπορούν να εκτελεστούν παράλληλα, θα ήταν αρκετές για να κρατούν απασχολημένο τον VLIW επεξεργαστή του προηγούμενου παραδείγματος. Αυτό όμως δεν συμβαίνει. Αρκετές από αυτές τις λειτουργικές μονάδες – μονάδες μνήμης, διακλάδωσης και κινητής υποδιαστολής – είναι σωληνωμένες και έχουν καθυστέρηση πολλαπλών κύκλων, απαιτώντας ένα μεγαλύτερο αριθμό λειτουργιών που μπορούν να εκτελεστούν παράλληλα ώστε να εμποδιστούν οι καθυστερήσεις.

Παράδειγμα

Αν η σωλήνωση κινητής υποδιαστολής έχει καθυστέρηση πέντε κύκλων και θέλουμε να προγραμματίσουμε μαζί δύο σωληνώσεις κινητής υποδιαστολής χωρίς καθυστερήσεις, τότε θα πρέπει να υπάρχουν 10 εντολές κινητής υποδιαστολής ανεξάρτητες από την πιο πρόσφατα διευθετημένη εντολή κινητής υποδιαστολής. Γενικά, χρειάζεται να βρούμε έναν αριθμό ανεξάρτητων εντολών σχεδόν ίσο με το μέσο βάθος σωλήνωσης, ίσο με τον αριθμό των λειτουργικών μονάδων. Αυτό σημαίνει ότι σχεδόν 15 ως 20 εντολές θα χρειαστούν για να κρατήσουμε απασχολημένο έναν επεξεργαστή πολλαπλής διευθέτησης με πέντε λειτουργικές μονάδες.

Οι πόροι του υλικού για έναν πολλαπλής διευθέτησης επεξεργαστή, αποτελεί το δεύτερο περιορισμό και προέρχεται από την ανάγκη για υλικό χρήσιμο στη διευθέτηση και στην εκτέλεση πολλαπλών εντολών ανά κύκλο. Είναι εύκολο να διπλασιάσουμε τις λειτουργικές μονάδες κινητής υποδιαστολής και ακεραίων. Όμως, υπάρχει μεγάλη αύξηση στο εύρος μνήμης και στο εύρος του αρχείου καταχωρητών. Ο VLIW επεξεργαστής απαιτεί έξι θύρες ανάγνωσης (δύο για κάθε φόρτωση αποθήκευση και δύο για το τμήμα των ακεραίων) και τρεις θύρες εγγραφής (μία για κάθε μη FP μονάδα) στο αρχείο καταχώρησης ακεραίων. Επίσης, απαιτεί έξι θύρες ανάγνωσης (μία για κάθε φόρτωση αποθήκευση και δύο για κάθε FP) και τέσσερις θύρες εγγραφής (μία για κάθε φόρτωση αποθήκευση ή FP) στο αρχείο καταχώρησης κινητής υποδιαστολής.

ΔΡΑΣΤΗΡΙΟΤΗΤΑ 4

Μπορείτε να προσδιορίσετε με ποιον τρόπο θα μπορούσαμε να επεκτείνουμε τον αριθμό των διευθετήσεων;

ΑΠΑΝΤΗΣΗ ΔΡΑΣΤΗΡΙΟΤΗΤΑΣ 4

Αν θέλουμε να επεκτείνουμε περισσότερο τον αριθμό των διευθετήσεων, θα πρέπει να προσθέσουμε επιπλέον υλικό και συγκεκριμένα θύρες μνήμης.

Οι ποινές της πολυπλοκότητας και του χρόνου πρόσβασης, της ιεραρχίας μιας μνήμης πολλαπλών θυρών, είναι πιθανότατα οι πιο σημαντικοί περιορισμοί που παρουσιάζονται σε οποιονδήποτε τύπο επεξεργαστή πολλαπλής διευθέτησης, είτε είναι υπερβαθμωτός είτε VLIW.

Ανάμεσα, στους υπερβαθμωτούς και στους VLIW επεξεργαστές υπάρχουν και αυτοί που χρησιμοποιούν συνδυασμό στατικού προγραμματισμού από τον μεταγλωττιστή και δυναμικού προγραμματισμού με χρήση υλικού, το οποίο παίρνει την απόφαση για το πόσες από τις επόμενες n εντολές θα διευθετήσει.

ΔΡΑΣΤΗΡΙΟΤΗΤΑ 5

Ποιες κατηγορίες επεξεργαστών πολλαπλής διευθέτησης γνωρίζετε και τι είδους προγραμματισμός χρησιμοποιείται σε αυτούς τους επεξεργαστές; Να συγκρίνετε την απάντησή σας με τους ορισμούς που υπάρχουν στην πρώτη υποενότητα: «Γενικά για τους VLIW και υπερβαθμωτούς επεξεργαστές».

Ανακεφαλαιώνοντας λοιπόν…

Οι επεξεργαστές πολλαπλής διευθέτησης επιτρέπουν σε πολλαπλές εντολές να διευθετούνται σε ένα κύκλο ρολογιού. Η μεγαλύτερη πρόκληση για όλους τους επεξεργαστές πολλαπλής διευθέτησης είναι να προσπαθήσουν να εκμεταλλευτούν όσο γίνεται τη παραλληλία σε επίπεδο εντολής.

Οι υπερβαθμωτοί επεξεργαστές διευθετούν μεταβαλλόμενο αριθμό εντολών σε κάθε κύκλο ρολογιού. Αυτοί είναι είτε στατικά προγραμματισμένοι από τον μεταγλωττιστή, είτε δυναμικά χρησιμοποιώντας τις τεχνικές του πίνακα αποτελεσμάτων και του Tomasulo.

Οι VLIW επεξεργαστές διευθετούν ένα καθορισμένο αριθμό εντολών, σε κάθε κύκλο ρολογιού, διαμορφωμένο είτε σαν μια μεγάλη εντολή, είτε σαν καθορισμένο πακέτο εντολών. Αυτοί είναι στατικά προγραμματισμένοι από τον μεταγλωττιστή.

Ο τρόπος οργάνωσης του επεξεργαστή, όπου οι μονάδες φόρτωσης και αποθήκευσης έχουν ουρές που επιτρέπουν σύνδεση με άλλες λειτουργικές μονάδες, ονομάζεται μη συζευγμένη αρχιτεκτονική.

Ανάμεσα, στους υπερβαθμωτούς και στους VLIW επεξεργαστές υπάρχουν και αυτοί που χρησιμοποιούν συνδυασμό στατικού προγραμματισμού από τον μεταγλωττιστή και δυναμικού προγραμματισμού με χρήση υλικού, το οποίο παίρνει την απόφαση για το πόσες από τις επόμενες n εντολές θα διευθετήσει.