5ο Εργαστήριο Αρχιτεκτονικής Η/Υ: Διοχέτευση Risc-V, πειραματισμός με τον Ripes

Α. Ευθυμίου

Παραδοτέο: Βλ. ecourse, 23:59

Με αυτή την εργαστηριακή άσκηση θα χρησιμοποιήσετε τον Ripes για να καταλάβετε πως λειτουργεί η διοχέτευση σε έναν επεξεργαστή. Θα πρέπει να έχετε μελετήσει τις διαλέξεις σχεδίασης επεξεργαστή μέχρι και την τεχνική της διοχέτευσης.

1 Ρυθμίσεις στον Ripes

Μέχρι τώρα χρησιμοποιήσαμε τον Ripes για εκτέλεση προγραμμάτων χωρίς να ενδιαφερόμαστε για την υλοποίηση, σε κυκλώματα, του επεξεργαστή. Γι'αυτό και στις ρυθμίσεις του Ripes (select processor - το εικονίδιο μικρού ολοκληρωμένου κυκλώματος, επάνω αριστερά) είχαμε επιλέξει Single-cycle processor. Σε αυτό το εργαστήριο, επιλέγουμε 5-stage processor. Στο Drop-down list Layout, συνήθως βολεύει η επιλογή Standard, αλλά κάποιες φορές είναι χρήσιμη και η επιλογή Extended, γιατί εμφανίζει χρήσιμες λεπτομέρειες. Τα παραπάνω αλλάζουν την όψη στο Processor tab (εικονίδιο μεγάλου ολοκληρωμένου κυκλώματος) και την συμπεριφορά του Ripes όταν εκτελούνται προγράμματα. Προσοχή η αλλαγή επεξεργαστή, ακόμα και αν αλλάξει απλά το Layout, ξεκινά την εκτέλεση προγράμματος από την αρχή.

Για πειραματισμό μπορείτε να δείτε και να συγκρίνετε τις διαφορές στην εκτέλεση προγραμματων με τις άλλες παραλλαγές του 5-stage processor όπου δεν έχει υλοποιηθεί κάποιο τμήμα ελέγχου προωθήσεων ή αναστολών εκτέλεσης (w/o xxxx).

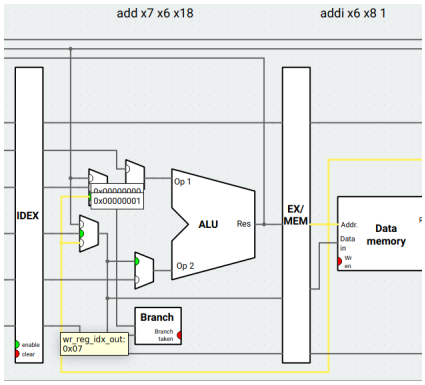
2 Λειτουργία Ripes με διοχετευμένο επεξεργαστή

Όταν έχει επιλεγεί ένας διοχετευμένος επεξεργαστής και προσομοιώνεται η εκτέλεση ενός προγράμματος, ο Ripes, στο Editor tab, δεν δείχνει (με κόκκινο) απλά την τρέχουσα εντολή, αλλά όλες τις εντολές που βρίσκονται σε κάποιο από τα στάδια διοχέτευσης. Αυτό καλύτερα να το παρατηρείτε στο μεσαίο παράθυρο που έχει της αληθινές εντολές μηχανής και όχι στο αριστερό που υπάρχουν και ψευτοεντολές assembly. Στα δεξιά κάθε εντολής φαίνεται το στάδιο στο οποίο βρίσκεται η εντολή σε αυτόν τον κύκλο.

Επίσης προσέξτε ότι, σε διοχετευμένο επεξεργαστή, η λειτουργία clock the circuit, που συμβολίζεται με το > (πλήκτρο F5), προχωράει την εκτέλεση κατά ένα στάδιο - κύκλο ρολογιού και όχι κατά μια εντολή. Αντίστοιχα, οι καταχωρητές ενημερώνονται από την εντολή που βρίσκεται στο τελευταίο στάδιο διοχέτευσης και η μνήμη από την εντολή που βρίσκεται στο στάδιο MEM (αν η εντολή προσπελαύνει μνήμη). Επομένως οι αλλαγές κατά την εκτέλεση του προγράμματος θα σας φαίνεται ότι έχουν κάποια καθυστέρηση.

Πιο ενδιαφέρον έχει όμως η παρατήρηση της εκτέλεσης στο processor tab (εικονίδιο μεγάλου ολοκληρωμένου κυκλώματος). Εκεί θα βρείτε σε block διάγραμμα όλα τα βασικά τμήματα του επεξεργαστή (σε μεγαλύτερη λεπτομέρεια αν έχετε επιλέξει το Extended Layout). Επάνω από το διάγραμμα εμφανίζονται οι εντολές που βρίσκονται σε κάθε στάδιο. Στα δεξιά υπάρχει η λίστα με τους καταχωρητές και τις τιμές τους, όπως και στο Editor tab. Κάτω δεξιά υπάρχει μια συνοπτική όψη της μνήμης δεδομένων. Φαίνεται ποια εντολή είναι σε κάθε στάδιο διοχέτευσης και οι επόμενες εντολές ώστε να μη χρειάζεται να αλλάζετε στο Editor tab για να δείτε ποιά εντολή θα προσκομιστεί στον επόμενο κύκλο. Η στήλη BP (breakpoint) δείχνει που υπάρχει ενεργό breakpoint όπου σταματάει η εκτέλεση για απασφαλμάτωση και παρατήρηση. Το Execution info έχει πληροφορίες για την ταχύτητα εκτέλεσης τους προγράμματος, τον αριθμο κύκλων από την αρχή της εκτέλεσης μέχρι τώρα, των εντολών που έχουν εκτελεστεί πλήρως (retired) και το CPI (cycles per instruction) και το αντίστροφό του, IPC.

Κάποια στοιχεια του διαγράμματος αλλάζουν δυναμικά όσο εκτελείται ένα πρόγραμμα. Επιλεγμένες είσοδοι πολυπλεκτών και σήματα ενεργοποίησης (enable σε καταχωρητές, Wr(ite) En(able) για το αρχείο καταχωρητών και τη μνήμη δεδομένων) εμφανίζονται με πράσινο. Αφήνοντας λίγη ώρα τον δείκτη ποντικιού (mouse hover) πάνω από μια είσοδο ή έξοδο, εμφανίζεται ένα βελάκι και το όνομα και η



Σχήμα 1: Προώθηση από μια αριθμητική εντολή στην αμέσως επόμενη.

τρέχουσα τιμή του αντίστοιχου σήματος. Με δεξί κλικ πάνω σε αυτά τα βελάκια μπορείτε να ενεργοποιήσετε το Show value που δείχνει συνεχώς την τρέχουσα τιμή. Έτσι μπορείτε να παρακολουθείτε την εκτέλεση του προγράμματος. Τα πιο σημαντικά σήματα για παρακολούθηση είναι ο PC, οι τιμές εισόδου και το αποτέλεσμα της ALU, η έξοδος της μνήμης δεδομένων και ο αριθμός καταχωρητή που αποθηκεύεται και η τιμή του. Ανάλογα με την εντολή όμως, μπορεί να χρειάζεται να βλέπετε και άλλα σήματα.

Πολύ χρήσιμο είναι και το διάγραμμα διοχέτευσης, Pipeline diagram, εικονίδιο που μοιάζει με πίνακα επάνω στη γραμμή εργαλείων του Ripes. Αυτό εμφανίζει το διάγραμμα διοχέτευσης μέχρι τον τρέχοντα κύκλο ρολογιού, με κάθε εντολή του προγράμματος να καταλαμβάνει μια γραμμή και το στάδιο στο οποίο βρισκόταν η εντολή στον κάθε κύκλο αναγράφεται στην αντίστοιχη στήλη. Με αυτό το διάγραμμα μπορεί να δει κανείς το ιστορικό της διοχετευμένης εκτέλεσης, πότε έγιναν καθυστερήσεις (stall), εκκενώσεις (flush). Δυστυχώς όσο το διάγραμμα είναι ανοιχτό, δεν μπορεί να συνεχιστεί η εκτέλεση του προγράμματος. Επομένως πρέπει να κλείνετε το παράθυρο του διαγράμματος διοχέτευσης μέχρι να φτάσετε στον κύκλο που σας ενδιαφέρει και να το ξανα-ανοίξετε.

3 Παραλαβή του σκελετού της άσκησης

Ακολουθήστε τον σύνδεσμο που υπάρχει στο ecourse για να δημιουργηθει το δικό σας αποθετήριο της άσκησης και κλωνοποιήστε το.

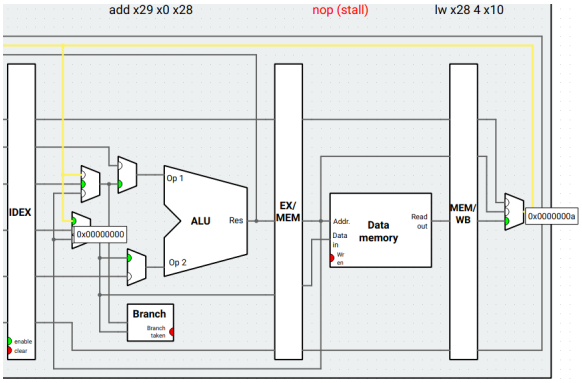
Στον κατάλογο που θα δημιουργηθεί, `lab05-ghUsername`, θα βρείτε το αρχείο `lab05.s`. Δεν υπάρχει python αρχείο για έλεγχο σε αυτή την άσκηση.

4 Επιβεβαίωση λειτουργίας διοχέτευσης

Ο υπάρχον κώδικας στο αρχείο `lab05.s` μπορεί να χρησιμοποιηθεί για να επιβεβαιωθεί ότι ο διοχετευμένος επεξεργαστής λειτουργεί σωστά σε αρκετές περιπτώσεις. Μετά τις 5 πρώτες εντολές που αρχικοποιούν καταχωρητές οι οποίοι χρησιμοποιούνται παρακάτω, οι επόμενες δύο επιβεβαιώνουν ότι προωθήσεις από μια εντολή που παράγει αποτέλεσμα στην ALU προς την αμέσως επόμενη λειτουργούν σωστά. Στο διάγραμμα του επεξεργαστή στον Ripes θα πρέπει να εμφανίζεται κάτι σαν την εικόνα 1. Με κίτρινο φαίνεται το μονοπάτι προώθησης και στον πολυπλέκτη που οδηγεί στην πρώτη είσοδο της ALU φαίνονται η (σωστή) τιμή που προωθείται και η (λάθος) τιμή που έρχεται από το αρχείο καταχωρητών.

Μετά από κάθε ακολουθία εντολών που έχουν σκοπό να δείξουν κάποιο χαρακτηριστικό σημείο της διοχέτευσης, ακολουθούν μερικές εντολές που δεν κάνουν τίποτα (γράφουν στον zero) ώστε να μην υπάρχουν κίνδυνοι δεδομένων που μπορεί να σας μπερδέψουν.

Στη δεύτερη ακολουθία εντολών υπάρχει μια load στον t3 που ακολουθείται από μια add που τον διαβάζει. Εδώ θα πρέπει να καθυστερήσει η add για έναν κύκλο και μετά η τιμή του t3 να προωθηθεί από την load, η οποία θα έχει φτάσει στο στάδιο write-back. Το σχήμα 2 φαίνεται η φυσαλίδα που



Σχήμα 2: Καθυστέρηση και προώθηση από μια load στην αμέσως επόμενη.

έχει βάλει η λογική ελέγχου διοχέτευσης μεταξύ των δύο εντολών (στο σταδιο MEM) και με κίτρινο το μονοπάτι προώθησης. Επίσης φαίνεται η (σωστή) τιμή που προωθείται (ως η έξοδος του πολυπλέκτη στο τέρμα δεξιά) και η (λάθος) τιμή που προέρχεται από το αρχείο καταχωρητών. Αν αφήσετε την εντολή add να προχωρήσει, θα δείτε ότι το αποτέλεσμα της είναι σωστό.

Ακολουθούν άλλα τρία παραδείγματα που δείχνουν πως η διοχέτευση αντιδρά σε άλματα και διακλαδώσεις.

Μπορείτε να αλλάξετε τον επεξεργαστή σε κάποια παραλλαγμένη εκδοχή όπου δεν έχουν υλοποιηθεί προωθήσεις ή έλεγχος για καθυστέρηση - stall - και εκεί θα δείτε ότι κάποιες εντολές δεν δίνουν σωστά αποτελέσματα.

5 Παραδοτέο

Αφού καταλάβετε καλά τον διοχετευμένο επεξεργαστή και πως φαίνεται στον Ripes, συμπληρώστε τον κώδικα του `lab05.s` με μερικά ακόμη παραδείγματα:

- Προσθέστε μια μικρή ακολουθία εντολών που δείχνει ότι αποτελέσματα αριθμητικών εντολών μπορούν να προωθηθούν σωστά στην μεθεπόμενη εντολή.
- Προσθέστε μια ακολουθία εντολών που ελέγχει αν ο επεξεργαστής προωθεί την πιο πρόσφατη τιμή σε περίπτωση που υπάρχει διπλός κίνδυνος: μια εντολή διαβάζει έναν καταχωρητή που γράφεται και από τις δύο προηγούμενες εντολές. Γράψτε τις προπορευόμενες εντολές έτσι ώστε να παράγουν δύο διαφορετικές τιμές, και δείτε ποιά προωθείται στην εντολή που διαβάζει τον καταχωρητή.
- Προσθέστε μια ακολουθία εντολών όπου μία load περνάει μια τιμή (μέσω καταχωρητή φυσικά) σε μια εντολή διακλάδωσης που δεν ακολουθείται. Βάλτε την εντολή διακλάδωσης αμέσως μετά την load. Θα πρέπει να δείτε την εντολή διακλάδωσης να καθυστερεί για ένα κύκλο και μετά να συνεχίζει κανονικά η εκτέλεση. Σκεφτείτε αν αυτός ο κύκλος καθυστέρησης είναι εξαιτίας κινδύνου δεδομένων ή κινδύνου ελέγχου.
- Προσθέστε ένα παράδειγμα όπου μια εντολή διακλάδωσης ακολουθείται και ο στόχος της (το label της εντολής) είναι η αμέσως επόμενη εντολή. Δείτε τί συμβαίνει στον Ripes και προσπαθήστε να καταλάβετε γιατί.

Παραδώστε το αλλαγμένο αρχείο στο αποθετήριο σας στο github.