5ο Εργαστήριο Αρχιτεκτονικής Η/Υ: Υλοποίηση λογικής προώθησης και καθυστέρησης διοχετευμένου επεξεργαστή

Α. Ευθυμίου

Παραδοτέο: Παρασκευή 4 Δεκέμβρη 2020, 23:59

Ο σκοπός αυτής της άσκησης είναι η εμβάθυνση της κατανόησης λειτουργίας ενός διοχετευμένου επεξεργαστή, χρησιμοποιώντας τα εργαλεία Quartus και Modelsim.

Σας δίνεται ένας διοχετευμένος επεξεργαστής, παρόμοιος με αυτόν του συγγράμματος. Υπάρχουν οι επεκτάσεις εντολών που είχαν γίνει και στον επεξεργαστή της προηγούμενης άσκησης (lui, ori, addi, addiu), εκτός της εντολής άλματος j. Ο επεξεργαστής έχει έτοιμη την «υποδομή» για προώθηση δεδομένων προς τα στάδια εκτέλεσης και προσπέλασης μνήμης, αλλά η λογική ελέγχου διοχέτευσης δεν έχει υλοποιηθεί και, συνεπώς, η λογική καθυστέρησης (stall) είναι συντηρητική αφού δεν επιτρέπονται ποτέ προωθήσεις. Στο πρώτο σκέλος της άσκησης (60% του βαθμού) χρησιμοποιώντας ένα σύντομο πρόγραμμα θα βρείτε και θα καταγράψετε τα σημεία όπου συμβαίνουν καθυστερήσεις λόγω κινδύνων δεδομένων ή ελέγχου και τη διάρκεια των καθυστερήσεων. Επιπρόσθετα, θα καταγράψετε τα σημεία όπου θα μπορούσε να γίνει προώθηση και πόσοι κύκλοι καθυστέρησης αποφεύγονται σε κάθε ένα. Στο δεύτερο σκέλος, η δουλειά σας είναι να γράψετε, σε VHDL, τη λογική προώθησης και καθυστέρησης ώστε να αποφεύγονται οι περισσότερες δυνατές καθυστερήσεις στον επεξεργαστή.

Για να κάνετε την άσκηση **είναι σημαντικό να έχετε μελετήσει τα μαθήματα για την υλοποίηση του MIPS σε πέντε στάδια διοχέτευσης** που αντιστοιχούν μέχρι την ενότητα 4.8 του συγγράμματος. Πρέπει επίσης να κάνετε μια γρήγορη επανάληψη στη γλώσσα VHDL και να θυμάστε βασικές αρχές ψηφιακής σχεδίασης.

1 Ο διοχετευμένος επεξεργαστής

Δημιουργείστε το αποθετήριό σας στο GitHub, ακολουθώντας το σύνδεσμο https://classroom.github.com/a/.

Ξεκινήστε το Quartus σύμφωνα με το σύστημά σας. Ανοίξτε το Quartus project με όνομα, mips.qpf, που υπάρχει έτοιμο στο αποθετήριο. Κάντε διπλό κλίκ στο mips για να δείτε το σχηματικό του επεξεργαστή. Εξερευνήστε το σχέδιο, δείτε τα περιεχόμενα των διαφόρων block/symbols και παρατηρήστε καλά τα ονόματα των σημάτων. **Μετατρέψτε το σχηματικό σε VHDL (File->Create/Update->Create HDL File From Current File), όπως και στην προηγούμενη άσκηση.**

Για να ξεχωρίζουν τα στάδια μεταξύ τους υπάρχουν σκούρες κόκκινες διακεκομμένες γραμμές επάνω και κάτω από κάθε καταχωρητή διοχέτευσης. Οι καταχωρητές διοχέτευσης είναι χρωματισμένοι με κίτρινο χρώμα. Οι ακροδέκτες εισόδου-εξόδου τους είναι χωρισμένοι σε ομάδες: επάνω βρίσκονται τα σήματα που χρησιμοποιούνται στο αμέσως-επόμενο στάδιο και πιο κάτω τα σήματα που απλά μεταφέρονται σε πιο μακρινά στάδια. Χαρακτηριστικός είναι ο καταχωρητής διοχέτευσης μεταξύ των σταδίων ID και EX, όπου υπάρχουν 3 τέτοιες ομάδες σημάτων / ζευγών ακροδεκτών: επάνω βρίσκονται αυτά που καταλήγουν στο EX, στη μέση αυτά που μεταφέρονται στο στάδιο MEM και κάτω αυτά που προορίζονται για το στάδιο WB (μέσω του MEM βέβαια).

Για να φαίνονται καλύτερα τα μονοπάτια προώθησης, τα καλώδια που χρειάζονται σε προηγούμενα στάδια (πηγαίνουν προς τα αριστερά), έχουν διαφορετικό χρώμα ανάλογα με το στάδιο από το οποίο προέρχονται. Αυτά του σταδίου WB είναι κόκκινα, του σταδίου MEM είναι σκούρα πράσινα, ενώ με μαύρο σημειώνεται η διεύθυνση επόμενης εντολής, σε περίπτωση διακλάδωσης.

Τα σήματα ελέγχου που είναι ενεργά σε κάποιο στάδιο έχουν γραμμές με ανοιχτό πράσινο χρώμα και το όνομα του σήματος είναι επίσης πράσινο. Σήματα ελέγχου που απλά μεταφέρονται σε άλλα στάδια έχουν το συνηθισμένο σκούρο μωβ του Quartus. Εξαίρεση αποτελούν τα σήματα καθυστέρησης, stall, και εκκένωσης, flush, που έχουν ρόζ χρώμα επειδή έχουν επίδραση σε αρκετά στάδια ταυτόχρονα.

Πολλά σήματα χρειάζονται και μεταφέρονται σε περισσότερα από ένα στάδια, μέσω των καταχωρητών διοχέτευσης. Για να έχουν ξεχωριστά ονόματα και να φαίνεται καθαρά η χρήση τους, τα ονόματα τους ξεκινούν με ένα προθεμα που καθορίζει το στάδιο στο οποίο αντιστοιχούν ενώ το υπόλοιπο μέρος του ονόματος περιγράφει τη σκοπιμότητα του σήματος. Για παράδειγμα, το σήμα ελέγχου, που καθορίζει αν ο πολυπλέκτης του σταδίου WB που επιλέγει μεταξύ της εξόδου της μνήμης (για lw) ή της εξόδου της ALU (για αριθμητικές-λογικές πράξεις), παράγεται στο στάδιο ID με το όνομα id_memToReg, συνεχίζει στο στάδιο EX με το όνομα ex_memToReg, μετά στο στάδιο MEM με όνομα mem_memToReg και, τέλος στο WB με όνομα wb_memToReg.

Με ροζ χρώμα έχει σημειωθεί και η μονάδα ελέγχου διοχέτευσης, pipe_ctrl, που παράγει τα σήματα προώθησης (id_forwardA, id_forwardB, id_ldstBypass), καθυστέρησης (stall) και εκκένωσης (flush). **Αν και στο σχηματικό υπάρχει πρόβλεψη (πολυπλέκτες) για προώθηση, προς το παρόν δεν γίνεται καμία προώθηση εκτός από την αυτόματη προώθηση μέσω του αρχείου καταχωρητών όταν ο ίδιος καταχωρητής ταυτόχρονα γράφεται στο στάδιο WB και διαβάζεται στο στάδιο ID.** Για το δεύτερο σκέλος της άσκησης θα αλλάξετε αυτή τη μονάδα ώστε να επιτρέπει τις υπόλοιπες προωθήσεις.

Στον διοχετευμένο επεξεργαστή της άσκησης ο υπολογισμός του στόχου διακλάδωσης γίνεται στο στάδιο ID με ξεχωριστό αθροιστή (br_adder), ενώ η απόφαση για τη διακλάδωση παίρνεται στο στάδιο EX, χρησιμοποιώντας την ALU που κάνει αφαίρεση. Αυτό έχει ως συνέπεια ότι κάθε διακλάδωση που ακολουθείται, ακυρώνει τις δύο εντολές που ακολουθούν στα στάδια ID, IF. Έτσι το σήμα flush χρησιμοποιείται και στο στάδιο ID και στο IF.

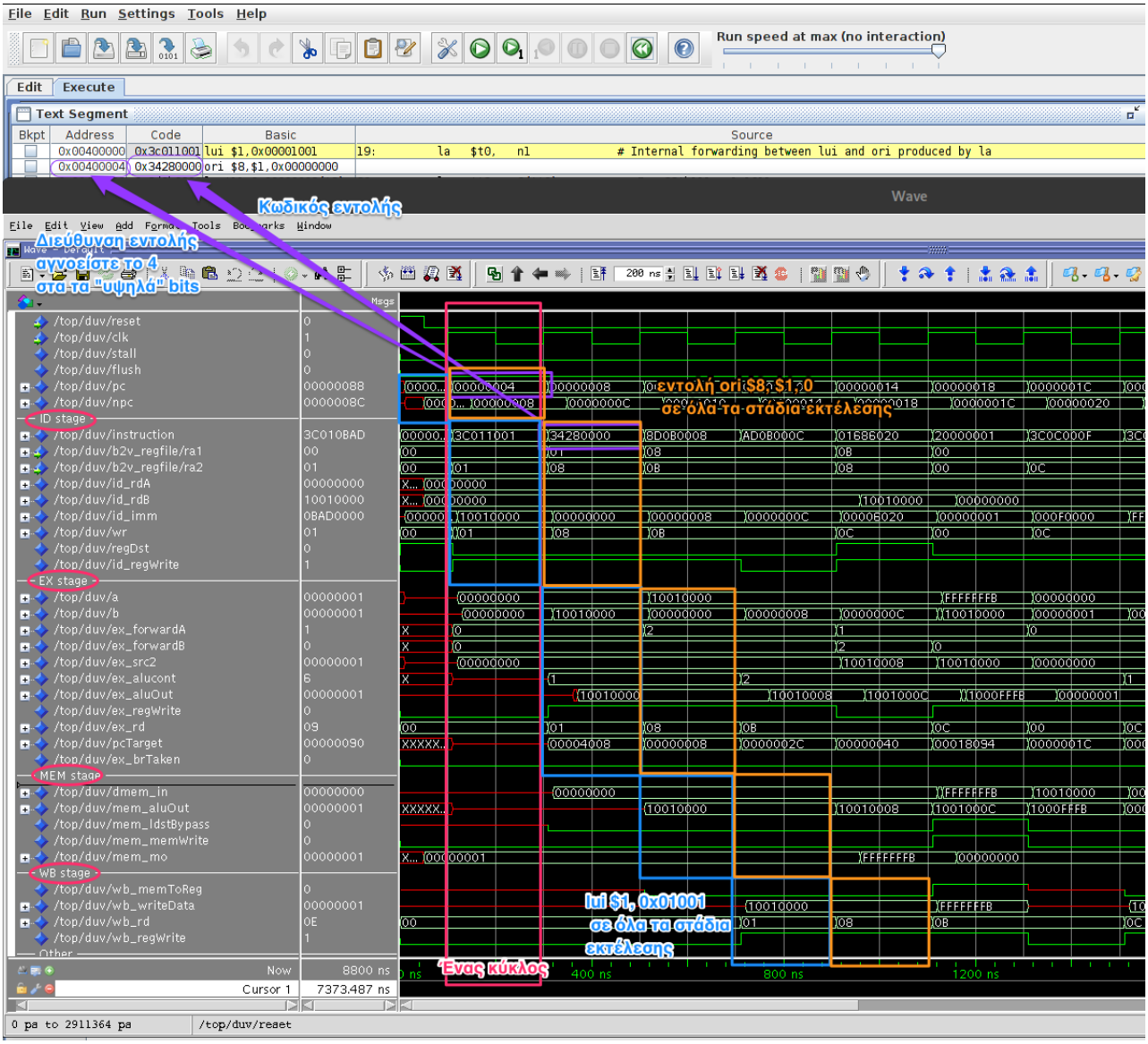
Υπάρχουν όμως και μερικές ακόμη αλλαγές:

- Υπάρχει πρόβλεψη για προώθηση μεταξύ διαδοχικών lw, sw που χρησιμοποιούν τον ίδιο καταχωρητή δεδομένων: συγκεκριμένα αν γίνεται αντιγραφή δεδομένων από τη μνήμη έτσι ώστε η lw να διαβάζει το δεδομένο και η sw να το γράφει ξανά στη μνήμη (σε άλλη θέση). Αυτό αναφέρεται ως «επιπλέον ανάπτυξη» στο 4.7, σελ. 433 του συγγράμματος και έχει εξηγηθεί στη διάλεξη. Για το σκοπό αυτό χρησιμοποιείται το σήμα ελέγχου ldStBypass και έχει προστεθεί ένας πολυπλέκτης στο στάδιο MEM (ldstbypass_mux), που δεν υπάρχει στις εικόνες του συγγράμματος. Αν το σήμα ελέγχου έχει την τιμή 1, τα δεδομένα που γράφονται στη μνήμη προέρχονται από το στάδιο WB όπου βρίσκεται η προηγούμενη lw.
- Η «μονάδα προώθησης» αντί να βρίσκεται στο στάδιο EX (ή στο MEM για προώθηση μεταξύ lw, sw που αναφέρεται παραπάνω), βρίσκεται στο στάδιο ID και αποτελεί μέρος του ελέγχου διοχέτευσης, του ρόζ μπλοκ με όνομα pipe_ctrl. Έτσι όλες οι αποφάσεις παίρνονται στο στάδιο ID και τα σήματα ελέγχου μεταφέρονται μέχρι το κατάλληλο στάδιο. Αυτό είναι για λόγους ομοιομορφίας με ό,τι συμβαίνει στα υπόλοιπα σήματα ελέγχου, όπως, για παράδειγμα, στο memToReg που αναφέρθηκε παραπάνω.

  Για παράδειγμα, στο πρόγραμμα:

  ```
  add $t0, ... # no dependencies from above
  or  $t1, $t0, $zero
  ```

  θα πρέπει να γίνει προώθηση της τιμής του `$t0` από το στάδιο MEM, όπου θα έχει φτάσει η add, στο στάδιο EX όπου θα έχει φτάσει η or. Η απόφαση αυτή όμως θα ληφθεί όταν η or βρίσκεται ακόμα στο στάδιο ID και η add στο στάδιο EX, γιατί ο έλεγχος γίνεται στο στάδιο ID. Κοιτώντας το σχηματικό του mips στο Quartus, η μονάδα αυτή θα παράγει την δυαδική τιμή `10`. (Αν η προώθηση γινόταν από την προ-προηγούμενη εντολή, η τιμή θα ήταν `01`, ενώ `00` σημαίνει ότι δεν γίνεται προώθηση). Μετά από ένα κύκλο η εντολή or θα βρεθεί στο στάδιο EX και το παραπάνω σήμα θα περάσει στο ex_forwardA και θα επιλέξει την προωθημένη τιμή του `$t0` από το στάδιο MEM (σήμα mem_aluOut, σκούρο πράσινο).
- Τέλος για απλοποίηση του σχηματικού και μεγαλύτερη ευελιξία, ο αποκωδικοποιητής εντολών της ALU (aludec στην υλοποίηση ενός κύκλου) έχει ενσωμματωθεί στον κύριο αποκωδικοποιητή (maindec).



Σχήμα 1: Κυματομορφή διοχετευμένου επεξεργαστή.

2 Κατανόηση προσομοίωσης διοχετευμένου επεξεργαστή

Για να προσομοιώσετε προγράμματα που τρέχουν στον επεξεργαστή θα πρέπει να χρησιμοποιήσετε τον Mars όπως και στην προηγούμενη άσκηση (με dump memory). Επειδή ο Mars κάνει dump περισσότερες λέξεις από την μνήμη δεδομένων, ανοίξτε με έναν editor το data_memfile.dat και σβήστε τις γραμμές από την 64 και μετά.

Επίσης πρέπει να ξεκινάτε την προσομοίωση με το Modelsim και να παρατηρείτε κυματομορφές, πάλι όπως στην προηγούμενη άσκηση. **Προσοχή όταν ξεκινήσετε το Modelsim, κλείστε το παλιό project που ανοίγει αυτόματα και, όταν ανοίξετε καινούριο, σιγουρευτείτε ότι έχετε βάλει τον σωστό κατάλογο, εκεί που έχετε κλωνοποιήσει το αποθετήριό σας, στο Project Location στο παράθυρο νέου project.** Επειδή πολλά ονόματα αρχείων είναι ίδια με το προηγούμενο εργαστήριο είναι εύκολο να μπερδευτείτε και να δουλεύετε σε λάθος κατάλογο. Ξεκινήστε ένα νέο project και προσθέστε σε αυτό

όλα τα αρχεία .vhd που βρίσκονται στον κατάλογο.

Η παρακολούθηση της εκτέλεσης είναι κάπως δυσκολότερη γιατί η εκτέλεση μιας εντολής γίνεται πλέον σε 5 κύκλους, υπάρχουν περισσότερα σήματα για να εξεταστούν και συμβαίνουν καθυστερήσεις (stall), εκκενώσεις (flush), κλπ. Το σχήμα 1 δείχνει 2 εντολές που εκτελούνται στον επεξεργαστή καθώς και τις αντιστοιχίσεις κώδικα μηχανής και διεύθυνσης εντολής μεταξύ MARS και κυματομορφής. Με μπλέ πλαίσιο φαίνεται η πρώτη εντολή (lui) καθώς προχωράει από στάδιο σε στάδιο σε κάθε κύκλο ρολογιού. Με πορτοκαλί φαίνεται η επόμενη εντολή (ori $8, $1, 0). Στα αριστερά φαίνονται τα όνόματα σημαντικών σημάτων ομαδοποιημένα ανά στάδιο διοχέτευσης.

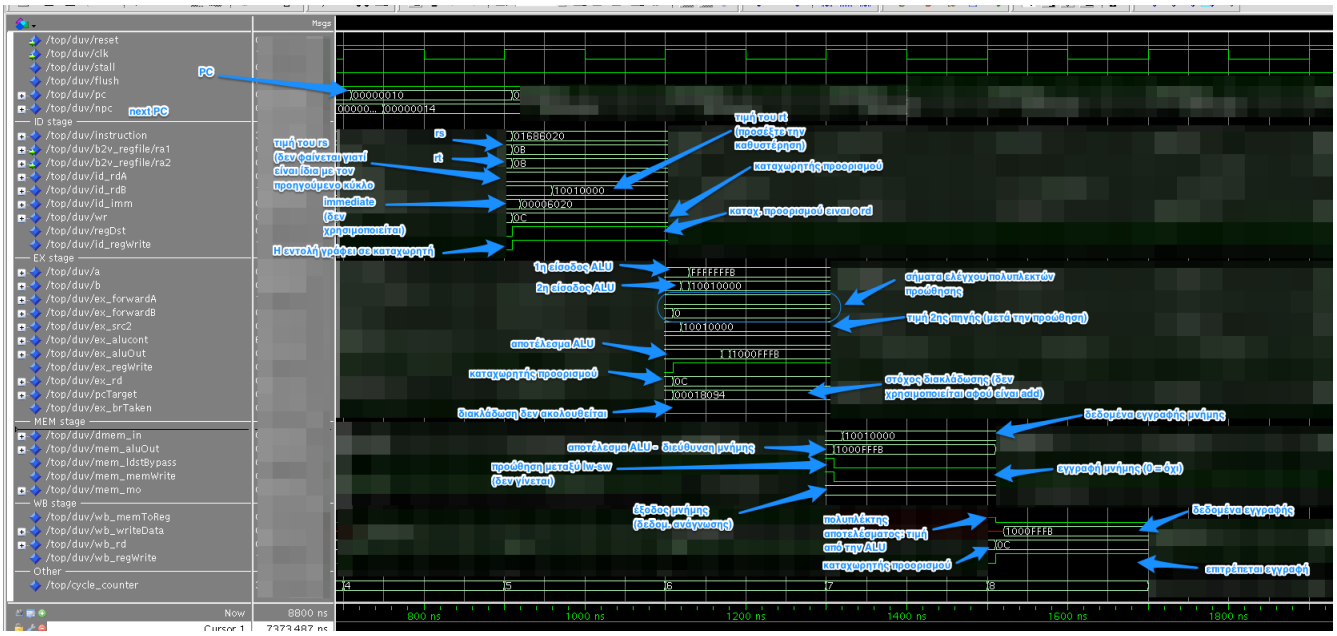
Ενώ στην προηγούμενη άσκηση όλα τα σήματα μιας εντολής φαινόταν στον ίδιο κύκλο, αυτό δεν συμβαίνει τώρα. Έτσι, όπως δείχνει το δεξί βελάκι του σχήματος 1, για να δεί κανείς την κωδικοποίηση της εντολής (σήμα instruction) πρέπει να περιμένει μέχρι αυτή να φτάσει στο 2ο κύκλο εκτέλεσής της, στο στάδιο ID.

Χρειάζεται προσοχή για να μη χαθεί κανείς σε μια τέτοια κυματομορφή. Η διεύθυνση της εντολής και η κωδικοποίησή της βοηθούν ώστε να αναγνωρίσει κανείς την εντολή που εκτελείται κάθε φορά.

Το σχήμα 2 δείχνει την κυματομορφή της εντολής `add $12, $11, $8)`. Τα σήματα πριν και μετά την εντολή έχουν σβηστεί για να φαίνεται καθαρότερα η εκτέλεσή της. Στο σχήμα επισημαίνονται τα σημαντικότερα σήματα. Θα πρέπει να δείτε το σχηματικό στο Quartus για τα υπόλοιπα σήματα.

Στον κώδικα VHDL του επεξεργαστή έχουν προστεθεί καθυστερήσεις σε βασικά κυκλώματα (π.χ. ALU, αρχείο καταχωρητών, μνήμες), όπως και στην προηγούμενη άσκηση. Έτσι σε κάθε ακμή του ρολογιού οι νέες τιμές εμφανίζονται με μια καθυστέρηση. Παρατηρήστε τις τιμές περίπου στην κατερχόμενη ακμή του ρολογιού, όταν οι τιμές θα έχουν σταθεροποιηθεί.

Ως βοηθήματα δίνονται ένα αρχείο wave.do που όταν φορτωθεί στο Modelsim παρακολουθεί τα σήματα που φαίνονται στα προηγούμενα σχήματα. Στο τέλος υπάρχει ένας μετρητής κύκλων εκτέλεσης, cycle_counter για να μπορείτε να γνωρίζετε σε ποιό σημείο της εκτέλεσης βρίσκεστε κάθε φορά.



Σχήμα 2: Κυματομορφή εντολής add $12,$11,$8.

3 Μέρος 1: προσομοίωση και εύρεση ευκαιριών προώθησης

Στο πρώτο μέρος θα χρησιμοποιήσετε το πρόγραμμα assembly labcode.asm. Τρέξτε με προσομοίωση αυτό το πρόγραμμα ως το τέλος και δείτε τι συμβαίνει σε κάθε κύκλο. Το τέλος του είναι όταν ολοκληρωθεί η αποθήκευση σε καταχωρητή (στάδιο WB) της τιμής της τελευταίας εντολής.

Θα πρέπει να βρείτε και να καταγράψετε τα σημεία όπου συμβαίνουν καθυστερήσεις λόγω κινδύνων δεδομένων ή ελέγχου και τη διάρκεια των καθυστερήσεων. Επιπρόσθετα, θα καταγράψετε τα σημεία όπου θα μπορούσε να γίνει προώθηση και πόσοι κύκλοι καθυστέρησης αποφεύγονται σε κάθε ένα. Οι απαντήσεις σε αυτό το μέρος της άσκησης θα υποβληθούν με ένα quiz στη σελίδα του μαθήματος στο ecourse. Ο απευθείας σύνδεσμος είναι http://ecourse.uoi.gr/mod/quiz/view.php?id=86760. Επιτρέπονται πολλαπλές προσπάθειες - αλλαγές μέχρι την προθεσμία.

4 Μερος 2: Ολοκλήρωση μονάδας ελέγχου διοχέτευσης

Η μονάδα ελέγχου διοχέτευσης παράγει τα σήματα ελέγχου που αφορούν τη διοχέτευση. Από αυτά, τα σήματα εκκένωσης, flush, που χρησιμοποιείται για να αδειάσει το προηγούμενο στάδιο της διοχέτευσης (IF), και αλλαγής ροής, flowChange, που χρησιμοποιείται για να επιλέξει την επόμενη τιμή του PC, είναι ήδη έτοιμα.

Τα σήματα που πρέπει να υλοποιήσετε είναι τα id_forwardA, id_forwardB, id_ldstBypass, και stall. Τα δύο πρώτα ελέγχουν τους πολυπλέκτες στις εισόδους της ALU που επιλέγουν μεταξύ της τιμής που προέρχεται από το αρχείο καταχωρητών, δηλαδή χωρίς να γίνει προώθηση, και τιμών που προέρχονται από τα στάδια MEM ή WB. Το id_ldstBypass ελέγχει τον πολυπλέκτη στην είσοδο δεδομένων προς εγγραφή στη μνήμη και επιλέγει μεταξύ της τιμής που προέρχεται από το αρχείο καταχωρητών ή την τιμή από την προηγούμενη lw (που θα βρίσκεται στο στάδιο WB).

Το σήμα stall, όταν είναι ενεργό (τιμή 1), εμποδίζει την τρέχουσα εντολή του σταδίου ID να προχωρήσει στο στάδιο EX και, φυσικά, την εντολή του σταδίου IF να προχωρήσει στο ID. Αν και υπάρχει ήδη μια υλοποίηση για το stall, είναι πολύ συντηρητική γιατί προς το παρόν δεν γίνονται καθόλου προωθήσεις. Θα πρέπει να το αλλάξετε ώστε αναβολές να συμβαίνουν μόνο όταν μια εντολή δεν έχει την τιμή που χρειάζεται επειδή αυτή προέρχεται από μια προηγούμενη lw.

Όλες οι αλλαγές θα γίνουν στο αρχείο pipe_ctrl.vhd. Ο κώδικας VHDL που θα γράψετε πρέπει να εξετάζει σε ποιές περιπτώσεις θα πρέπει να γίνει προώθηση και από πού καθώς και σε ποιές περιπτώσεις θα γίνει καθυστέρηση. Θα χρειαστείτε μερικές if-else, πιθανότατα με σύνθετες συνθήκες (`or`, `and` ...), αλλά σίγουρα δεν θα χρειαστεί να κάνετε επαναλήψεις ή κάτι ιδιαίτερα περίπλοκο. Δείτε την υπάρχουσα υλοποίηση του stall ως ένα παράδειγμα υλοποίησης τέτοιων σημάτων.

Η μονάδα ελέγχου διοχέτευσης δέχεται ως εισόδους πολλά σήματα. Κάποια προέρχονται από άλλα στάδια και μεταφέρουν τον καταχωρητή προορισμού του σταδίου και πληροφορίες για το αν η εντολή στο στάδιο είναι φόρτωση από μνήμη ή αν η εντολή πράγματι γράφει αποτελέσματα σε καταχωρητή. Πολλά σήματα προέρχονται από τον κύριο αποκωδικοποιητή που βρίσκεται στο στάδιο ID και προσδιορίζουν («εξηγούν») την τρέχουσα εντολή. Δεν θα πρέπει να χρειαστείτε άλλα σήματα (θύρες) εισόδου-εξόδου. Επειδή στα σήματα ελέγχου οι λεπτομέρειες είναι εξαιρετικά σημαντικές, μερικές φορές, αντί να στηριχθείτε στα ονόματα σημάτων για να καταλάβαιτε τί κάνουν, ίσως να χρειαστεί να δείτε τον κώδικα άλλων modules, και ιδιαίτερα του κύριου αποκωδικοποιητή (maindec).

Για βαθμολόγηση θα χρησιμοποιηθεί ένα πιο αναλυτικό πρόγραμμα από το labcode.asm που εξετάζει περισσότερες περιπτώσεις. Αυτό δεν σας παρέχεται, γιατί θα μπορούσατε διαβάζοντάς το να βρείτε τις περιπτώσεις που πρέπει να εξετάσετε στον έλεγχο διοχέτευσης!

4.1 Παραδοτέο

Το παραδοτέο του δεύτερου μέρους της άσκησης είναι το αλλαγμένο αρχείο `pipe_ctrl.vhd`. Αν ελέγχοντας την ορθότητα της υλοποίησης προωθήσεων γράψατε κάποιο πρόγραμμα assembly που εξετάζει ενδιαφέρουσες περιπτώσεις, μπορείτε να το προσθέσετε και αυτό, με όνομα αρχείου lab05.asm.

Η παράδοση θα γίνει μέσω GitHub, όπως πάντα. **Όπως και στο προηγούμενο παραδοτέο μήν ανε-**

βάσετε στο GitHub κανένα άλλο αρχείο, εκτός αυτών που υπάρχουν στο αρχικό αποθετήριο, γιατί πιάνουν πολύ χώρο.

5 Καθαρισμός αρχείων

Στους υπολογιστές των εργαστηρίων, επειδή ο διαθέσιμος χώρος σας στο δίσκο είναι περιορισμένος (quota), και τα εργαλεία που χρησιμοποιήσατε δημιουργούν πολλά και μεγάλα αρχεία, όταν τελειώσετε με την άσκηση, σβήστε όλα τα περιτά αρχεία. Κρατήστε μόνο ότι αρχείο υπάρχει ήδη στο αποθετήριο του GitHub και τυχόν προγράμματα assembly που γράψατε για έλεγχο.