6ο Εργαστήριο Αρχιτεκτονικής Η/Υ: Προσομοίωση και επέκταση επεξεργαστή

Α. Ευθυμίου

Παραδοτέο: Παρασκευή 24 Απρίλη 2015, 23:00

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

Σας δίνεται ένας ολόκληρος επεξεργαστής σχεδόν ίδιος με αυτόν του συγγράμματος (συμπεριλαμβανομένης της εντολής jump), που παρουσιάστηκε στις διαλέξεις. Πρώτα θα γράψετε ένα πρόγραμμα σε γλώσσα μηχανής που χρησιμοποιεί όλες τις υλοποιημένες εντολές του επεξεργαστή ώστε να βεβαιωθείτε ότι δουλεύει σωστά. Μετά θα προσθέσετε μερικές επιπλέον εντολές και θα τις ελέγξετε επίσης. Στην επόμενη εργαστηριακή άσκηση θα τροποποιήσετε τον επεξεργαστή ώστε να γίνει διοχετευμένος.

Θα πρέπει να έχετε μελετήσει τα μαθήματα για την υλοποίηση του MIPS σε ένα κύκλο ρολογιού που αντιστοιχούν μέχρι την ενότητα 4.4 του βιβλίου. Πρέπει επίσης να κάνετε μια γρήγορη επανάληψη στη γλώσσα Verilog και να θυμάστε βασικές αρχές ψηφιακής σχεδίασης.

Μή ξεχάσετε να επιστρέψετε τα παραδοτέα που αναφέρονται στο τέλος του κειμένου για να πάρετε βαθμό γι'αυτή την εργαστηριακή άσκηση!

1 Η άσκηση

Για να ξεκινήσετε θα χρειαστείτε όλα τα αρχεία του εργαστηρίου δίνοντας τις εξής εντολές:

```
git remote add lab06_starter https://github.com/UoI-CSE-MYY402/lab06_starter.git
git fetch lab06_starter
git merge lab06_starter/master -m "Fetched lab06 starter files"
```

Ξεκινήστε το Quartus σύμφωνα με το σύστημά σας. Ανοίξτε το Quartus project με όνομα: mips.qpf 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, γιατί θα τα χρειαστείτε για την μετέπειτα προσομοίωση.

2 Μετατροπή προγραμμάτων σε γλώσσα μηχανής

Για να προσομοιώσετε τον επεξεργαστή θα χρειαστείτε προγράμματα σε γλώσσα μηχανής και αρχικές τιμές στη μνήμη δεδομένων. Τα modules dmem και imem έχουν τη δυνατότητα να φορτώσουν τα περιεχόμενα των αρχείων `data_memfile.dat` και `instr_memfile.dat` στις αντίστοιχες μνήμες. Αν ανοίξετε τα αρχεία αυτά θα δείτε δεκαεξαδικούς αριθμούς.

Για να δημιουργήσετε τα δικά σας αρχεία θα χρησιμοποιήσετε το Mars. Ανοίξτε το αρχείο lab06.asm στον Mars:

```
   .data
number:
   .word  15

   .globl main

   .text
main:
    add $t1, $zero, $zero
```

Παρατηρήστε ότι υπάρχει μόνο μια γραμμή κώδικα και ότι λείπει το συνηθισμένο τέλος προγραμμάτων

assembly με την εντολή syscall. Αυτό γίνεται γιατί δεν είναι υλοποιημένη αυτή η syscall στον επεξεργαστή. Το ίδιο πρέπει να κάνετε και στα υπόλοιπα προγράμματά που θα τρέχουν στον επεξεργαστή.

Μετατρέψτε το σε γλώσσα μηχανής πατώντας το κουμπί "Assemble" (F3).

Τώρα μπορείτε να πάρετε το αντίστοιχο πρόγραμμα σε γλώσσα μηχανής, αλλά σε 2 χωριστά αρχεία για εντολές και δεδομένα. Από το μενού File, επιλέξτε Dump Memory. Θα εμφανιστεί ένα μικρό παράθυρο. Μπορείτε να επιλέξετε μεταξύ ".text" για το πρόγραμμα και ".data" για τα δεδομένα. Η μορφή που μπορεί να διαβαστεί από τη Verilog είναι "Hexadecimal Text" Τα ονόματα αρχείων πρέπει να είναι `data_memfile.dat` για δεδομένα και `instr_memfile.dat` για εντολές. Μη χρησιμοποιήσετε άλλα ονόματα. Τα imem.v, dmem.v περιμένουν τα συγκεκριμένα αρχεία για να δουλέψουν.

Για κάποιο λόγο (bug) το αρχείο δεδομένων που γράφει το Mars περιέχει 1024 γραμμές. Ο προσομοιωτής αργότερα διαμαρτύρεται, αλλά η προσομοίωση γίνεται κανονικά.

3 Προσομοίωση του επεξεργαστή

Αφού έχετε δημιουργήσει τα αρχεία που φορτώνονται στις μνήμες, μπορείτε να προσομοιώσετε τον επεξεργαστή.

Ξεκινήστε τον προσομοιωτή με την εντολή vsim. Αν το Modelsim «θυμάται» το παλιό modelsim project του lab05, κλείστε το με File > Close. Το Modelsim μπορεί να θυμάται και τον προηγούμενο κατάλογο από όπου το τρέξατε, οπότε προσέξτε σε διάφορα παράθυρα/διαλόγους μήπως και δεν έχετε δώσει το σωστό κατάλογο.

Ξεκινήστε ένα νέο modelsim project με όνομα `mips_sim`. Προσθέστε όλα τα αρχεία Verilog (.v) **εκτός από** τα `lpm_constant4_bb.v lpm_constant4_syn.v`.

Κάντε Compile > Compile All

Ξεκινήστε την προσομοίωση: Simulate > Start Simulation. Διαλέξτε το top.v από τη βιβλιοθήκη Work.

Επιλέξτε τα σήματα που θέλετε να παρακολουθήσετε στις κυματομορφές. Θα πρέπει να κοιτάτε το σχηματικό στο Quartus για να βρίσκετε τα ονόματα των σημάτων που θέλετε. Για ευκολία, σας δίνεται ένα αρχείο με τα κύρια σήματα: wave.do. Για να το χρησιμοποιήσετε, επιλέξτε File > Load και στο παράθυρο που θα εμφανιστεί διαλέξτε το παραπάνω αρχείο. Μπορείτε να αλλάξετε θέση στα σήματα τραβώντας τα με το ποντίκι. Επίσης μπορείτε να τοποθετείτε διαχωριστικά (όπως στο wave.do) με Add > Divider

Τρέξτε την προσομοίωση, με το κατάλληλο εικονίδιο ή το F9. Για το παραπάνω πρόγραμμα της μίας εντολής, αρκούν 300ps. Κάντε Zoom-full (μεγεθυντικός φακός με σκούρο εσωτερικό ή πλήκτρο F), για να δείτε καλύτερα τις τιμές των σημάτων.

Παρατηρήστε ποιοι καταχωρητές διαβάζονται, τις τιμές τους, την πράξη που κάνει η ALU, το αποτέλεσμα, τον αριθμό καταχωρητή στον οποίο γράφεται το αποτέλεσμα, κλπ. Βεβαιωθείτε ότι δουλεύει σωστά.

Στην επόμενη ακμή του ρολογιού, οι περισσότερες κυματομορφές κοκκινίζουν γιατί παίρνουν τιμές X, την ειδική τιμή της Verilog που δείχνει ότι δεν γνωρίζει αν ένα bit είναι 1 ή 0. Αυτό συμβαίνει γιατί η επόμενη εντολή που διαβάζεται από τη μνήμη (PC = 4) δεν είναι καθορισμένη και παριστάνεται με X. Ακολούθως τα X εξαπλώνονται σχεδόν σε όλα τα σήματα. Αυτό είναι φυσιολογικό να συμβαίνει στο τέλος του προγράμματος, αλλά αν, σε πιο σύνθετα προγράμματα, το δείτε να συμβαίνει κατά τη διάρκεια της εκτέλεσης, κάποιο λάθος θα πρέπει να έχει συμβεί.

4 Βήμα 1: συγγραφή και προσομοίωση προγράμματος ελέγχου

Τώρα που γνωρίζετε τη διαδικασία συγγραφής προγράμματος, μετατροπής του σε γλώσσα μηχανής και προσομοίωσης, ήρθε η στιγμή να γράψετε ένα μικρό προγραμμα που επαληθεύει τη λειτουργία του επεξεργαστή. Αλλάξτε το lab06.asm για το σκοπό αυτό.

Για την ώρα ο επεξεργαστής υλοποιεί το υποσύνολο των εντολών MIPS: add, sub, and or, lw, sw, beq, j.

Οπότε θα πρέπει να γράψετε ένα μικρό πρόγραμμα που περιλαμβάνει όλες αυτές τις εντολές. Επιπλέον θα πρέπει να υπάρχουν τουλάχιστον 2 εκτελέσεις beq: μία που ακολουθείται (branch taken) και μία που δεν ακολουθείται, ώστε να δοκιμαστούν και οι δύο περιπτώσεις. Επίσης, θα πρέπει να χρησιμοποιήσετε τιμές διαφορετικές από το 0 σε όσες περισσότερες περιπτώσεις γίνεται.

Αν χρειάζεστε σταθερές μπορείτε να τις αποθηκεύσετε στη μνήμη δεδομένων. Οι καταχωρητές, εκτός του 0 (zero) **δεν είναι αρχικοποιημένοι και περιέχουν X**. Μη στηρίζεστε στο γεγονώς ότι έχουν 0 όπως στον Mars.

Μη ξεχνάτε όμως ότι δεν έχουμε διαθέσιμες τις (ψευτο-)εντολές la, li. Οπότε για να διαβάσετε από τη μνήμη, υπολογίστε τη διεύθυνση «με το χέρι» και χρησιμοποιείστε εντολές σαν την `lw $t0, 8($zero)` για να διαβάσετε την 3η λέξη της μνήμης δεδομένων. Η μνήμη δεδομένων χρησιμοποιεί μόνο τα λιγότερο σημαντικά bit της διεύθυνσης, επομένως μπορείτε να θεωρήσετε ότι ξεκινάει από τη διεύθυνση 0 και όχι από τη 0x10010000 που χρησιμοποιεί ο Mars. Παρόμοια και για τη μνήμη εντολών.

Όταν ολοκληρώσετε την προσομοίωση, πάρτε ένα screenshot το παράθυρο με τις κυματομορφές που να δείχνει ότι λειτουργεί σωστά ο επεξεργαστής με όλες τις εντολές. Ονομάστε το αρχείο όπως θέλετε αρκεί να φαίνεται εύκολα ότι είναι για την πρώτη προσομοίωση.

Δεν θα πρέπει να υπάρχουν λάθη στον επεξεργαστή. Αν βρείτε κάποιο λάθος και είστε σίγουροι ότι δεν φταίει το προγραμμά σας, ειδοποιήστε τον διδάσκοντα!

5 Βήμα 2: Επέκταση του επεξεργαστή

Στο βήμα αυτό θα επεκτείνετε τον επεξεργαστή ώστε να εκτελεί μερικές ακόμη εντολές, συγκεκριμένα τις: addi, andi, ori, slti. Οι εντολές αυτές χρειάζονται αλλαγές μόνο στα κυκλώματα ελέγχου.

Θεωρούμε ότι η σταθερά immediate χρειάζεται επέκταση προσήμου και όχι επέκταση με 0[1]. Συνεπώς ο υπάρχοντας δίαλος δεδομένων (datapath) είναι ικανός να τις εκτελέσει χωρίς αλλαγές.

Θα χρειαστεί λοιπόν να αλλάξετε μόνο το αρχείο maindec.v. Επειδή δεν θα αλλάξετε κανένα σχηματικό, για να προσομοιώσετε τον αλλαγμένο επεξεργαστή θα χρειαστεί να επαναλάβετε μόνο τα βήματα από το compile All και μετά.

6 Βήμα 3: Επαλήθευση του επεκταμένου επεξεργαστή

Χρησιμοποιώντας ως βάση το lab06.asm που γράψατε στο βήμα 1, γράψτε ένα πρόγραμμα με όνομα lab06v2.asm που επαληθεύει τη λειτουργία των 4 νέων εντολών που προσθέσατε.

Επειδή θα χρειαστεί να αλλάξετε τα αρχεία `data_memfile.dat` `instr_memfile.dat` για να προσομοιώσετε τον επεκταμένο επεξεργαστή, **κρατήστε τα προηγούμενα (του δικού σας lab06.asm) με κατάλληξη .old**.

Όταν ολοκληρώσετε την προσομοίωση, πάρτε ένα screenshot το παράθυρο με τις κυματομορφές που να δείχνει ότι λειτουργεί σωστά ο επεξεργαστής με όλες τις εντολές. Ονομάστε το αρχείο όπως θέλετε, αρκεί να φαίνεται εύκολα ότι είναι για την δεύτερη προσομοίωση.

7 Παραδοτέα

Τα παραδοτέα της άσκησης είναι τα αρχεία `lab06.asm`, `data_memfile.old` `instr_memfile.old`, `main_dec.v`, `lab06v2.asm`, `data_memfile.dat` `instr_memfile.dat` και τα δύο screenshots των κυματομορφών. Η παράδοση θα γίνει μέσω GitHub, όπως πάντα.

8 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 υπάρχει στο αποθετήριο του GitHub (μερικά από αυτά θα πρέπει να τα έχετε αλλάξει) το lab06v2.asm και τα screenshots.

[1] Αυτό δεν είναι σωστό γιατί ο MIPS ορίζει ότι οι andi, ori επεκτείνουν τη σταθερά με 0.