ΗΥ-225: Οργάνωση Υπολογιστών
Άνοιξη 2006

Τμ. Επ. Υπολογιστών

Σειρά Ασκήσεων 13: Υλοποίηση Εξαιρέσεων και Τελικές Διορθώσεις

Προθεσμία έως Κυριακή 14 Μαΐου 2006, ώρα 23:59 (βδομάδα 11-12)

13.1 Εξαιρέσεις Μνήμης:

Εκτός από τις εξαιρέσεις παράνομης εντολής (illegal opcode) --που έχει και ο επεξεργαστής του βιβλίου (παράγραφος 5.6, σελίδες 410-416) και που θα υλοποιήσουμε και εμείς-- και εκτός από τις εξαιρέσεις αριθμητικής υπερχείλισης --που δεν θα υλοποιήσουμε εμείς-- σε αυτή τη σειρά ασκήσεων θα υλοποιήσετε και Εξαιρέσεις Προσπελάσεων Μνήμης:

- **Παραβίαση Ευθυγράμμισης** (Alignment Violation): Στη μνήμη του MIPS, όλες οι λέξεις (32 bits) πρέπει να έχουν διεύθυνση που να είναι ακέραιο πολλαπλάσιο του 4, δηλ. τα 2 LS bits να είναι μηδέν (περιορισμός ευθυγράμμισης). Το δικό μας υποσύνολο του MIPS προσπελαύνει πάντα λέξεις (32 bits) όποτε προσπελαύνει τη μνήμη, και αυτό το κάνει σε 3 περιπτώσεις:
  i. ανάγνωση εντολής (i_fetch),
  ii. ανάγνωση δεδομένων (4ος κύκλος της lw), ή
  iii. εγγραφή δεδομένων (4ος κύκλος της sw).

  Κατά τη διάρκεια αυτών των τριών καταστάσεων (κύκλων), θα πρέπει να κάνετε το κύκλωμα ελέγχου να παρακολουθεί τη διεύθυνση **ma** που δίδεται στη μνήμη: εάν τα 2 LS bits αυτής της διεύθυνσης δεν είναι 00, τότε θα πρέπει να προκαλείται εξαίρεση, δηλαδή ο έλεγχος να μεταβαίνει, σαν επόμενη κατάσταση, στην ειδική κατάσταση εξαιρέσεων "exception_st" που θα πούμε παρακάτω. Παρατηρήστε ότι, όταν γίνει αυτή η μετάβαση, η "κακιά" προσπέλαση μνήμης θά έχει ήδη γίνει, αλλά θεωρούμε ότι η μνήμη έχει αρκετή δική της προστασία ώστε ή να αρνηθεί να εκτελέσει την παράνομη προσπέλαση ή να αγνοήσει τα 2 μη μηδενικά LS bits της διεύθυνσης.
- **Σφάλμα Σελίδας Μνήμης** (Page Fault): Σ' έναν πραγματικό υπολογιστή MIPS με εικονική μνήμη (virtual memory), όπως θα μάθουμε αργότερα, ορισμένες διευθύνσεις μνήμης αντιστοιχούν σε "σελίδες" (pages) που βρίσκονται στην κύρια μνήμη, ενώ άλλες όχι. Κάθε προσπέλαση σε απούσα σελίδα μνήμης προκαλεί μιάν εξαίρεση γνωστή σαν "σφάλμα σελίδας" (page fault). Στον δικό μας επεξεργαστή των ασκήσεων 10-12, η μνήμη έχει μόνο 1024 λέξεις, στις χαμηλές διευθύνσεις, άρα γιά μας σφάλμα σελίδας θα είναι:
  i. κάθε προσπέλαση σε διεύθυνση μνήμης μεγαλύτερη ή ίση του 4096 (1024 λέξεις = 4096 bytes), και
  ii. κάθε προσπέλαση στη διεύθυνση μνήμης μηδέν (0): ο επεξεργαστής μας θα θεωρεί σφάλμα σελίδας κάθε τέτοια προσπέλαση προκειμένου να μην περνάν απαρατήρητες οι παράνομες προσπελάσεις μνήμης μέσω NULL pointers (άρα, διεύθυνση εκκίνησης του PC δεν μπορεί να είναι η 0).

  Τα σφάλματα σελίδας μπορούν να εμφανιστούν στις ίδιες καταστάσεις (κύκλους) όπως και οι παραβιάσεις ευθυγράμμισης που είδαμε παραπάνω, και μάλιστα μπορεί να εμφανιστούν και ταυτόχρονα με εκείνες. Έτσι, η ανίχνευση και η αντίδραση θα είναι κοινή γιά αυτούς τους δύο τύπους εξαιρέσεων. Παρατηρήστε και πάλι ότι η "κακιά" προσπέλαση μνήμης θα έχει ήδη γίνει όταν θα μεταβούμε στην κατάσταση "exception_st", αλλά θεωρούμε ότι η μνήμη έχει αρκετή δική της προστασία ώστε να αρνηθεί να εκτελέσει την προσπέλαση όταν η διεύθυνση αναφέρεται σε απούσα λέξη.

Άσκηση 13.2: Προσθήκη Εξαιρέσεων (Exceptions)

Προσθέστε στο datapath και στον έλεγχο όσα χρειάζονται γιά να υλοποιεί ο επεξεργαστής σας **Εξαιρέσεις Παράνομης Εντολής** και **Εξαιρέσεις Μνήμης**. Η ιδέα είναι η ίδια με ό,τι περιγράφει το βιβλίο στην παράγραφο 5.6 (σελίδες 410-416), αλλά εμείς θα έχουμε διαφορετικές αιτίες εξαιρέσεων, και μία μόνο πρόσθετη κατάσταση ελέγχου, την "**`exception_st`**", γιά να τις υλοποιήσουμε. Όπως και στο βιβλίο, πάντως, θα προσθέσετε δύο νέους (ακμοπυροδότητους) καταχωρητές: "**epc**" (Exception PC), των 32 bits, γιά να κρατά τη διεύθυνση της "κακιάς" εντολής (συν 4), και "**cause**" (αιτία), επίσης των 32 bits, γιά να κρατά έναν κώδικα που υποδεικνύει την αιτία της εξαίρεσης. Περιγράψτε τον "epc" σαν RegLd (όπως και τον pc), ενώ τον "cause" μπορείτε να τον περιγράψτε σαν reg --όπως και τον currState. Οι αιτίες εξαιρέσεων και οι αντίστοιχες δράσεις θα είναι:

- **Παράνομη Εντολή** (Illegal Instruction): Στη διάρκεια του 2ου κύκλου κάθε εντολής, δηλαδή κατά την κατάσταση **decode_rr**, εάν ο έλεγχος διαπιστώσει ότι ο συνδυασμός op-funct δεν αντιστοιχεί σε καμία από τις εντολές που ο επεξεργαστής μας ξέρει να εκτελεί (αυτές των ασκήσεων 11 και 12.1), φορτώνει τον αριθμό **μηδέν (0)** στον καταχωρητή cause, και μεταβαίνει, σαν επόμενη κατάσταση, στην κατάσταση exception_st.
- **Εξαιρέσεις Μνήμης** (Memory Exceptions): Όπως είπαμε παραπάνω, στη διάρκεια των κύκλων (καταστάσεων) προσπέλασης μνήμης, **i_fetch**, **mem_rd**, και **mem_wr**, το κύκλωμα ελέγχου θα παρακολουθεί τη διεύθυνση ma που δίδεται στη μνήμη, και εάν αυτή δεν είναι σωστή φορτώνει στον καταχωρητή cause τον αριθμό που δίδεται παρακάτω (στο δεκαδικό), και μεταβαίνει, σαν επόμενη κατάσταση, στην κατάσταση exception_st.
    5. Παραβίαση ευθυγράμμισης κατά τη διάρκεια i_fetch.
    6. Σφάλμα σελίδας κατά τη διάρκεια i_fetch.
    7. Παραβίαση ευθυγράμμισης **και** σφάλμα σελίδας κατά τη διάρκεια i_fetch.
    9. Παραβίαση ευθυγράμμισης κατά τη διάρκεια mem_rd.
    10. Σφάλμα σελίδας κατά τη διάρκεια mem_rd.
    11. Παραβίαση ευθυγράμμισης **και** σφάλμα σελίδας κατά τη διάρκεια mem_rd.
    13. Παραβίαση ευθυγράμμισης κατά τη διάρκεια mem_wr.
    14. Σφάλμα σελίδας κατά τη διάρκεια mem_wr.
    15. Παραβίαση ευθυγράμμισης **και** σφάλμα σελίδας κατά τη διάρκεια mem_wr.
- **exception_st:** Στην κατάσταση αυτή πρέπει πρώτον να αποθηκεύσουμε τη διεύθυνση της "κακιάς" εντολής (συν 4) στον καταχωρητή **epc**, και δεύτερον να φορτώσουμε στον **pc** τη διεύθυνση όπου ξεκινά ο interrupt handler. Η διεύθυνση της "κακιάς" εντολής (συν 4) είναι το (ήδη αυξημένο κατά 4) περιεχόμενο του pc. Η διεύθυνση όπου ξεκινά ο interrupt handler, στο μεν κανονικό MIPS είναι η 32'h80000080, στη δε δική μας περίπτωση θα την κάνουμε να είναι η **32'd2048**, προκειμένου αυτή να πέφτει μέσα στην υλοποιημένη μνήμη (στη μέση της).

Άσκηση 13.3: Πλήρες Debugging

Ελέγξτε εξονυχιστικά τον επεξεργαστή που σχεδιάσατε, και διορθώστε τα λάθη μέχρι ότου αυτός εκτελεί επιτυχώς τα μεγάλα προγράμματα που θα βρείτε στην περιοχή "**`~hy225/verilog/test/test13/`**". Υπολογίστε με τη βοήθεια του SignalScan πόση είναι η ελάχιστη δυνατή περίοδος του ρολογιού σας, και αλλάξτε το top-level ώστε να τρέχει ο επεξεργαστής σας με αυτό το **γρηγορότερο δυνατό ρολόϊ**. Πόσα MHz καταφέρατε να "πιάσετε";

Παραδώστε, σχεδόν όπως και στις προηγούμενες ασκήσεις, τον κώδικά σας, "**datapath13.v**" και "**control13.v**", το νέο test bench και μνήμη, "**test13.v**" και "**memory13.hex**", και ένα χαρακτηριστικό στιγμιότυπο, "**signals13.jpg**", από το Signalscan με ένα από τα πλέον πολύπλοκα tests, πακεταρισμένα σε ένα αρχείο, που όμως αυτή τη φορά να είναι τύπου **zip** και όχι tar όπως πριν (προκειμένου να αποφεύγεται καλύτερα ο κίνδυνος καταστροφής του πρώτου αρχείου πηγής αν ξεχάσει κανείς το όρισμα προορισμού), "**`ask13.zip`**", μέσω:

```
zip  ask13.zip   datapath13.v control13.v test13.v memory13.hex signals13.jpg
~hy225/bin/submit 13
```

Προαιρετική Άσκηση 13.4: Απλή Ομοχειρία (Προ-Ανάγνωση επόμενης Εντολής)

Όσοι από σας ενδιαφέρεστε ιδιαίτερα γιά το hardware και έχετε χρόνο, αλλάξτε το datapath και την FSM και το κύκλωμα ελέγχου προκειμένου να εισάγετε στον επεξεργαστή σας μιάν απλή μορφή ομοχειρίας (pipelining) δύο (μόνο) επιπέδων: κατά τη διάρκεια εκτέλεσης κάθε εντολής θα γίνεται και η ανάγνωση της επόμενης εντολής. Αυτή η απλή μορφή pipelining ονομάζεται συνήθως "fetch-execute overlap".

Στην παλαιά (non-pipelined) μορφή του επεξεργαστή, κάθε εντολή "παραδίδει" στην επόμενή της --εκτός από τα περιεχόμενα των καταχωρητών γενικού σκοπού $0-$31 και τα περιεχόμενα της μνήμης-- την τιμή (διεύθυνση) που περιέχεται στον PC, και η οποία καθορίζει από πού θα διαβαστεί η επόμενη εντολή. Στη νέα (pipelined) μορφή του επεξεργαστή, κάθε εντολή θα "παραδίδει" στην επόμενή της --εκτός από τους $0-$31 και τη μνήμη-- τόσο τη διεύθυνση της επόμενης εντολής **συν 4**, σαν περιεχόμενο του PC, όσο **και την ίδια την επόμενη εντολή**, σαν περιεχόμενο του IR. Έτσι, η εκτέλεση της **κάθε** εντολής θα ξεκινά με **έτοιμο, φορτωμένο** τον καταχωρητή IR, και επομένως θα γλιτώνουμε τον ένα κύκλο ανάγνωσης της εντολής, δηλαδή **δεν θα υπάρχει η κατάσταση i_fetch**. Αυτό θα επιτευχθεί ώς εξής:

- **decode_rr:** Όλες οι εντολές ξεκινούν με την κατάσταση αυτή. Στη διάρκειά της, εκτός από όσα ήδη γίνονταν, θα γίνονται και τα εξής:
  - Διαβάζουμε από M[pc] και φυλάμε αυτό που διαβάσαμε στον (νέο) καταχωρητή irNxt. Αφού ο καταχωρητής pc περιέχει τη διεύθυνση της παρούσας εντολής **συν 4**, αυτό που διαβάσαμε από τη μνήμη είναι η "από κάτω" εντολή. Σε περίπτωση που η παρούσα εντολή αποδειχθεί ότι δεν είναι εντολή επιτυχημένης μεταφοράς ελέγχου, αυτό που διαβάσαμε θα είναι η επόμενη εντολή (next IR).
  - Υπολογίζουμε την τιμή pc+4, και την φυλάμε σ' έναν (νέο) καταχωρητή pcNxt. Δεδομένου ότι η κεντρική ALU είναι απασχολημένη με τον υπολογισμό του pc+Immx4, χρειαζόμαστε έναν δεύτερο αθροιστή (ALU). Σε περίπτωση που η παρούσα εντολή αποδειχθεί ότι δεν είναι εντολή επιτυχημένης μεταφοράς ελέγχου, αυτό που υπολογίσαμε θα είναι ο PC (συν 4) της επόμενης εντολής.
- **jump:** Στην κατάσταση αυτή διαβάζουμε από M[jmpAddr] και φυλάμε αυτό που διαβάσαμε στον ir (είναι η επόμενη εντολή). Επίσης, υπολογίζουμε την τιμή jmpAddr+4 και την φυλάμε στον pc (είναι η διεύθυνση της επόμενης εντολής συν 4).
- **branch:** Είτε προσθέτετε μιά νέα κατάσταση μετά από την κατάσταση αυτή, και εκεί πηγαίνετε όταν η διακλάδωση επιτύχει και κάνετε τα ανάλογα όπως και παραπάνω στην jump, είτε μέσα στην κατάσταση branch διαβάζετε από M[ALUout] (εντολή προορισμού) και υπολογίζετε το ALUout+4, και στο τέλος του κύκλου φορτώνετε τον ir από M[ALUout] (επιτυχημένη διακλάδωση) ή από irNxt (αποτυχημένη διακλάδωση), και τον pc ανάλογα από ALUout+4 ή pcNxt.
- **Τελευταίος κύκλος των υπολοίπων εντολών:** επιπλέον του ό,τι κάνατε σε κάθε τέτοιον τελευταίο κύκλο, φορτώνετε τον ir από τον irNxt και τον pc από τον pcNxt, ώστε να τα βρεί έτοιμα η κατάσταση decode_rr της επόμενης εντολής.

Up to the Home Page of CS-225

last updated: 4 May 2006, by M. Katevenis.