1ο Εργαστήριο Αρχιτεκτονικής Η/Υ: MIPS assembly, γνωριμία με τον MARS

Α. Ευθυμίου

Παραδοτέο: Παρασκευή 27 Φλεβάρη, 16:00

Με αυτή την εργαστηριακή άσκηση θα εξοικειωθείτε με τον MARS έναν προσομοιωτή γλώσσας assembly του MIPS, που θα χρησιμοποιήσετε σε πολλές επόμενες εργαστηριακές ασκήσεις. Θα πρέπει να έχετε μελετήσει το πρώτο μάθημα για τη γλώσσα assembly του MIPS που αντιστοιχεί στις ενότητες 2.1-2.3 του βιβλίου.

Μή ξεχάσετε να επιστρέψετε, μέσω GitHub, το παραδοτέο που αναφέρεται στο τέλος του κειμένου για να πάρετε βαθμό γι'αυτή την εργαστηριακή άσκηση!

Χρησιμοποιείστε το Piazza για ανταλλαγή συμβουλών, πληροφοριών, καλών πρακτικών χρήσης κτλ, αλλά μή δίνετε πληροφορίες που φανερώνουν τη λύση των ασκήσεων. Οι βοηθοί και ο διδάσκοντας θα συμμετέχουν στις συζητήσεις αυτές.

Ο MARS, που αναπτύχθηκε από τους Pete Sanderson και Ken Vollmar, εκτός απο προσομοιωτής είναι ένα πλήρες γραφικό περιβάλον ανάπτυξης και εξσφαλμάτωσης προγραμμάτων (IDE). Στο εργαστήριο 0 δόθηκαν οδηγίες για την εγκατάστασή του. Επειδή είναι γραμμένος σε Java τρέχει σε υπολογιστές με οποιοδήποτε λειτουργικό σύστημα αρκεί να υπάρχει εγκατεστημένη η κατάλληλη έκδοση του Java J2SE. Στους υπολογιστές των εργαστηρίων του Τμήματος θα τον βρείτε στο `~myy402/bin/Mars4_5.jar`.

1 Βασικές πληροφορίες για MIPS assembly

- Τα προγράμματα assembly αποθηκεύονται σε απλά αρχεία κειμένου και οι καταλήξεις τους είναι συνήθως `.asm` ή `.s`.
- Τα προγράμματα καλό είναι να έχουν μία ετικέτα (**label**) με το όνομα `main:` που δηλώνει από που θα ξεκινήσει η εκτέλεση. Στον MARS χρειάζεται και η δήλωση `.globl main` στην αρχή του προγράμματος. Στις εργαστηριακές ασκήσεις πρέπει απαραίτητα να υπάρχει η ετικέτα main. Δείτε παρακάτω μία ρύθμιση του MARS που πρέπει να κάνετε ώστε τα προγράμματά σας να ξεκινούν από το main.
- Τα προγράμματα πρέπει να τελειώνουν με τις εξής δύο εντολές: `addi $v0,$0,10` και `syscall`. Με αυτές, όταν ολοκληρωθεί η εκτέλεση του προγράμματος, ο έλεγχος περνάει ξανά στο λειτουργικό σύστημα.
- Τα σχόλια ξεκινούν με το σύμβολο # και τελειώνουν στο τέλος της γραμμής.
- Δεν επιτρέπεται να γράψετε πάνω από μία εντολή ανά γραμμή.
- Οι ετικέτες (labels) πρέπει να τελειώνουν με το σύμβολο :
- Ο assembler, το πρόγραμμα που διαβάζει κώδικα assembly, είναι εξαιρετικά απλός. Μπορεί να μην επιτρέπει για παράδειγμα οι εντολές να «σπάνε» σε ξεχωριστές γραμμές και τα μηνύματα λάθους μπορεί να μην είναι πάντα αρκετά κατατοπιστικά. Ακολουθήστε το πρότυπο των προγραμμάτων που σας δίνονται για να γράφετε σωστό κώδικα που ακολουθεί το στυλ με το οποίο είναι εξοικειωμένοι οι περισσότεροι προγραμματιστές στον κόσμο. Οι βασικές οδηγίες είναι:
  - οι ετικέτες γράφονται τέρμα αριστερά και είναι σχετικά μικρές.
  - οι εντολές ξεκινούν δεξιότερα για να ξεχωρίζουν από τις ετικέτες και είναι στοιχισμένες.
  - οι τελεσταίοι των εντολών επίσης απέχουν μερικά κενά από τα ονόματα των εντολών.
- Ο assembler, εκτός από εντολές assembly, ετικέτες και σχόλια, δέχεται και κάποιες ειδικές λέξεις ως οδηγίες που λέγονται directives. Χρησιμοποιούνται για να ξεχωρίσουν το πρόγραμμα από τα δεδομένα και για να δεσμεύσουν χώρο στη μνήμη για δεδομένα. Τα directives ξεκινούν πάντα με μία τελεία. Θα μάθετε μερικές directives από το πρώτο πρόγραμμα assembly παρακάτω.

2 Εξοικείωση με τον MARS

Κάντε τα παρακάτω βήματα:

1. Το repository σας στο GitHub, έχει τη μορφή: <όνομα χρήστη GitHub>-labs. Για την ώρα είναι άδειο. Παρόλα αυτά, κλωνοποιείστε το, στον υπολογιστή σας (εργαστήριο ή σπίτι ή και τα δύο), αγνοώντας το σχετικό μήνυμα του git. Δείτε τις σχετικές οδηγίες για το git από το 2ο μάθημα και το φυλλάδιο του εργαστηρίου 0.

   Για να πάρετε τα αρχεία της εργαστηριακής άσκησης, μεταβείτε στον κατάλογο που θα δημιουργηθεί από το παραπάνω βήμα και θα έχει το ίδιο όνομα με το αποθετήριο. (cd <όνομα χρήστη GitHub>-labs). Μετά, κάνετε τα παρακάτω βήματα:

   ```
   git remote add lab01_starter https://github.com/UoI-CSE-MYY402/lab01_starter.git
   git fetch lab01_starter
   git merge lab01_starter/master -m "Fetched lab01 starter files"
   ```

   Θα δείτε ότι θα εμφανιστεί ένας κατάλογος lab01. Μεταβείτε σε αυτόν: cd lab01. Εκεί θα βρείτε το αρχείο `lab01.asm`, το πρώτο σας πρόγραμμα σε MIPS assembly.

2. Ξεκινήστε τον MARS, π.χ. σε τερματικό με την εντολή `java -jar <path_to_Mars4_5.jar>`.

3. Στο μενού Settings ενεργοποιήστε το "Initialize Program Counter to global main", ώστε η εκτέλεση να ξεκινά από την ετικέτα main. Επίσης, επειδή δε θα χρειαστείτε τους co-processors, μπορείτε να σείρετε το δεξί υπό-παράθυρο προς τα δεξιά ώστε να φαίνεται μόνο το tab "Registers". Έτσι θα υπάρχει περισσότερος χώρος για να βλέπετε τον κώδικα, κυρίως κατά την εκτέλεσή του (στο tab "execute"). Αν τα γράμματα σας φαίνονται πολύ μικρά, μπορείτε να αλλάξετε το μέγεθός τους (και ό,τι άλλο θέλετε) από τις ρυθμίσεις στο Settings→Editor.

4. Φορτώστε το αρχείο `lab01.asm` χρησιμοποιώντας είτε το εικονίδιο (2ο από αριστερά) ή από το μενού: File→Open, ή, ακόμα καλύτερα, με τη συντόμευση από το πληκτρολόγιο (Ctrl-O).

   Διαβάστε προσεκτικά τον κώδικα (και τα σχόλια) στο "Edit" tab. Παρατηρήστε ότι οι εντολές, ετικέτες και σχόλια χρωματίζονται αυτόματα (code highlighting) και ότι υπάρχει η δυνατότητα αυτόματης συμπλήρωσης εντολών (completion suggestion).

5. Όταν πλέον καταλαβαίνετε καλά τον κώδικα, ετοιμάστε τον για εκτέλεση (αγγλικός όρος assemble) χρησιμοποιώντας το μενού (Run→Assemble), ή το εικονίδιο με το κατσαβίδι και το κλειδί ή με το πλήκτρο F3. Αυτόματα θα αλλάξει ο MARS στο tab "Execute", όπου φαίνονται 3 παράθυρα στη θέση του κώδικα: η μνήμη εντολών (text segment), η μνήμη δεδομένων (data segment) και η αντιστοίχιση ετικετών σε διευθύνσεις (labels). Στα δεξιά θα παραμείνει το παράθυρο με τους καταχωρητές. Παρατηρήστε τις πληροφορίες στα παράθυρα. Ο αρχικός κώδικας (και τα σχόλια) φαίνονται ακόμη μαζί με άλλες πληροφορίες όπως η διεύθυνση κάθε εντολής και η κωδικοποίησή της σε γλώσσα μηχανής (που θα είναι το αντικείμενο επόμενου μαθήματος). Στο παράθυρο δεδομένων μπορεί κανείς να δεί τα δεδομένα σε δύο διαφορετικές μορφές.

6. Εκτελέστε το πρόγραμμα εντολή προς εντολή χρησιμοποιώντας το Run→Step ή το κατάλληλο εικονίδιο ή πλήκτρο σύντμευσης. Παρατηρήστε τις αλλαγές στους καταχωρητές και στη μνήμη. Το MARS δείχνει με χρώμα τις τελευταίες αλλαγές. Σε ένα σημείο σας ζητείται να δώσετε τον αριθμό μητρώου σας (matriculation number). Αυτό γίνεται στο ύπο-παράθυρο στο κάτω μέρος. Πρέπει να πατήσετε Return/Enter για να δεχθεί την είσοδο ο MARS.

7. Εξερευνήστε όλες τις επιλογές, τουλάχιστον, του μενού Run και κυρίως τις συντμεύσεις πλήκτρων. Θα χρησιμοποιείσετε το MARS σε πολλές εργαστηριακές ασκήσεις: όσο περισσότερο γνωρίζετε τις δυνατότητές του τόσο πιο παραγωγικοί θα είστε, συνεπώς θα τελειώνετε τις ασκήσεις γρηγορότερα.

3 Ερωτήσεις αυτοεξέτασης

Για να βεβαιωθείτε ότι εκτελέσατε σωστά την άσκηση, θα πρέπει να μπορείτε να απαντήσετε τις παρακάτω ερωτήσεις. Προσπαθήστε μόνοι σας. Μη ξεχνάτε ότι το MARS έχει ένα ενσωματομένο εγχειρίδιο (μενού Help ή πλήκτρο F1). Αν κολλήσετε, ρωτήστε τους συμφοιτητές σας αυτοπροσώπως ή στο Piazza, αλλά όχι για οτιδήποτε σχετίζεται με το παραδοτέο παρακάτω.

1. Ποιά είναι η διεύθυνση του var3, και ποιά του array+0x10;
2. Πώς μπορείτε να διαβάσετε το string που είναι αποθηκευμένο στο mesg1 στο παράθυρο δεδομένων;
3. Γιατί είναι γραμμένο κάπως ανάποδα;
4. Πώς μπορείτε να αλλάξετε την τιμή της var1 χωρίς να αλλάξετε τον κώδικα από το tab "Edit";
5. Μπορείτε να αλλάξετε τιμές καταχωρητών κατά την εκτέλεση;
6. Ποιές είναι οι διαφορές των αποτελεσμάτων εκτέλεσης των lb και lbu;

4 Αυτόματος έλεγχος ορθότητας

Στο repository θα βρείτε έναν κατάλογο που λέγεται test. Εκεί μέσα υπάρχει ένας μηχανισμός αυτόματου ελέγχου των αποτελεσμάτων του προγράμματός σας. Ο έλεγχος γίνεται με σύγκριση τιμών καταχωρητών και διευθύνσεων μνήμης με προϋπολογισμένες τιμές που θα πρέπει να δίνει ένα σωστό πρόγραμμα. Αυτό θα χρησιμοποιηθεί και από το διδακτικό προσωπικό ώς ένας από τους τρόπους ελέγχου και βαθμολόγησης.

Για την ώρα το πρόγραμμα που σας δόθηκε δεν περνάει τον έλεγχο. Αν το τρέξετε με το υπάρχον lab01.asm αρχείο, θα πάρετε μηνύματα λάθους γιατί ο ελεγκτής δεν μπορεί να χειριστεί είσοδο από το πληκτρολόγιο, πρέπει το πρόγραμμα assembly να μπορεί να εκτελεστεί χωρίς διακοπές.

Για τον έλεγχο χρησιμοποιείται η γλώσσα Ruby και, επομένως, αν χρησιμοποιήσετε δικό σας υπολογιστή για την άσκηση, θα πρέπει να εγκαταστήσετε τη Ruby. Δείτε το φυλλάδιο του εργαστηρίου 0 για οδηγίες. Θα χρειαστεί επίσης να αλλάξετε το αρχείο `lab01_tester`, στη γραμμή 17, ώστε να μπορεί να βρεί το αρχείο Mars4_5.jar.

Το αρχείο `mips_tester.rb` είναι αυτό που τρέχει τον MARS και καταγράφει την έξοδο της προσομοίωσης και τις τιμές καταχωρητών και μνήμης που μας ενδιαφέρουν. Δε χρειάζεται να κάνετε καμιά αλλαγή σε αυτό.

Στο αρχείο `lab01_tester`, στη γραμμή που ξεκινά με τη λέξη `expect`, βρίσκονται οι καταχωρητές, οι θέσεις μνήμης και οι σωστές τελικές τιμές που πρέπει να έχουν. Για καταχωρητές η μορφή είναι `:<register name> => <expected value>` Για τη μνήμη η μορφή είναι `'<address>' => <expected value>`. Οι διευθύνσεις και οι αναμενόμενες τιμές είναι σε δεκαεξαδική μορφη και ξεκινούν με 0x. Ως μέρος του παραδοτέου αυτής της άσκησης θα δώσετε τις δικές σας αναμενόμενες τιμές.

5 Παραδοτέο

Αλλάξτε τις γραμμές 23-34 του lab01.asm ώστε να φορτώνουν από τη μνήμη τον αριθμό μητρώου σας αντί να ζητούν να διαβαστεί από το πληκτρολόγιο. Αποθηκεύστε τον αριθμό μητρώου σας στη μνήμη γράφοντάς τον στη θέση της ετικέτας matric: χρησιμοποιώντας τον editor του MARS, αντικαταστήστε το 0 με τον αριθμό μητρώου σας στην παραπάνω ετικέτα. Μη ξεχάσετε να ξανακάνετε assemble το πρόγραμμά σας αφού τώρα πια έχει αλλάξει.

Αφού δοκιμάσετε ότι τρέχει σωστά, αλλάξτε τις αναμενόμενες τιμές στο αρχείο `test/lab01_tester` και τρέξτε το lab01_tester από τον κατάλογο test, χρησιμοποιώντας ένα τερματικό.

Μη ξεχάσετε να κάνετε commit τις αλλαγές σας και να τις στείλετε (push) στο GitHub repository για να βαθμολογηθούν πριν από την καταληκτική ημερομηνία!