2ο Εργαστήριο Αρχιτεκτονικής Η/Υ: MIPS assembly: Πίνακες και επαναλήψεις

Α. Ευθυμίου

Παραδοτέο: Τρίτη 1 Μάρτη 2016, 23:00

Το αντικείμενο αυτής της άσκησης είναι ένα πρόγραμμα που διατρέχει τα στοιχεία ενός πίνακα από ακέραιους και υπολογίζει δύο αποτελέσματα. Θα πρέπει να έχετε μελετήσει τα μαθήματα για τη γλώσσα assembly του MIPS που αντιστοιχούν μέχρι και την ενότητα 2.7 του βιβλίου (4η διάλεξη).

Σε αρχικό μάθημα εξηγήθηκε η σημασία της ακαδημαϊκής δεοντολογίας. Ο κώδικας που θα παραδώσετε θα πρέπει να είναι αποτέλεσμα ατομικής εργασίας. Τα προγράμματα που θα παραδοθούν θα ελεγχθούν για ομοιώτητα με ειδικό λογισμικό. Επιτρέπεται, και είνα θεμιτή, η συνεργασία και ανταλλαγή πληροφοριών σε πρακτικά θέματα, για παράδειγμα σύνταξη εντολών assembly, λύση προβλημάτων σχετικών με το git, αλλά όχι η συνεργασία στη συγγραφή κώδικα.

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

1 Περιγραφή του αλγορίθμου

Το πρόγραμμα θα υπολογίζει το άθροισμα των περιττών, θετικών αριθμών και το άθροισμα των άρτιων, αρνητικών αριθμών που βρίσκονται σε έναν πίνακα. Ο πίνακας περιέχει ακέραιους αριθμούς των 32bit με κωδικοποίηση συμπληρώματος ως προς 2. Το πρόγραμμα θα πρέπει να διατρέχει όλα τα στοιχεία του πίνακα, εξετάζοντας αν το κάθε στοιχείο θα πρέπει να προστεθεί σε ένα από τα δύο αθροίσματα. Συνεπώς μόνο ένα πέρασμα των στοιχείων του πίνακα επιτρέπεται.

Τα τελικά αποτελέσματα θα αποθηκεύονται ως εξής:

- στον καταχωρητή `$s0` το άθροισμα των περιττών, θετικών αριθμών.
- στον καταχωρητή `$s1` το άθροισμα των άρτιων, αρνητικών αριθμών.

Ο πίνακας θα είναι αποθηκευμένος στη μνήμη και το μέγεθός του επίσης θα δίνεται σε μια θέση μνήμης. Το μέγεθος του πίνακα, μπορεί να είναι 0, δηλαδή να μην περιέχει στοιχεία, οπότε το αποτέλεσμα του προγράμματος σε αυτή την περίπτωση θα πρέπει να είναι 0 και για τα δύο αθροίσματα.

Δεν επιτρέπεται η χρήση εντολών διαίρεσης και υπολογισμού υπόλοιπου. Θα πρέπει να βρείτε με άλλο τρόπο αν ένας αριθμός είναι άρτιος ή περιττός. Επίσης δεν επιτρέπεται η χρήση ψευτοεντολών εκτός από τις li, la. Η αλλαγμένη έκδοση του MARS που χρησιμοποιείται στο εργαστήριο δεν τις δέχεται, επομένως όποια εντολή περνάει από τον `MarsMYY402_4_5.jar` (εκτός των διαιρέσεων) μπορείτε να τη χρησιμοποιήσετε.

Τέλος, μπορείτε να αγνοήσετε φαινόμενα υπερχείλισης. Δεν θα μας απασχολήσουν σε αυτή την άσκηση.

2 Σκελετός προγράμματος

Για να πάρετε τα αρχεία της εργαστηριακής άσκησης, μεταβείτε στον κατάλογο εργασίας που είχατε κλωνοποιήσει από το αποθετήριό σας στο GitHub. (cd <όνομα χρήστη GitHub>-labs). Μετά, κάνετε τα παρακάτω βήματα:

```
git remote add lab02_starter https://github.com/UoI-CSE-MYY402/lab02_starter.git
git fetch lab02_starter
git merge lab02_starter/master -m "Fetched lab02 starter files"
```

Θα δείτε ότι θα εμφανιστεί ένας κατάλογος lab02. Μεταβείτε σε αυτόν: `cd lab02`. Εκεί θα βρείτε το αρχείο `lab02.asm`, που περιέχει το σκελετό του προγράμματος.

Θα δείτε ότι περιέχει λίγες γραμμές που:

- διαβάζουν τον αριθμό στοιχείων του πίνακα στον καταχωρητή `$a0` (γραμμές 12-13)
- διαβάζουν την αρχική διεύθυνση του πίνακα (base address) στον καταχωρητή `$a1` (γραμμή 15)
- αρχικοποιούν τα δύο αθροίσματα με την τιμή 0 στους καταχωρητές `$s0, $s1` (γραμμές 17-18)
- τελειώνουν την εκτέλεση με ένα syscall (γραμμές 27-28)

Τα σχόλια δείχνουν σε ποιό σημείο θα γράψετε τον δικό σας κώδικα.

3 MARS Debugging

Τώρα που τα προγράμματά σας γίνονται μεγαλύτερα και πιο πολύπλοκα, ο απλοϊκός τρόπος αποσφαλμάτωσης με εκτέλεση εντολή προς εντολή, δεν είναι πλέον αποδοτικός. Αν αντιμετωπίζετε ένα πρόβλημα που εμφανίζεται σχετικά αργά στην εκτέλεση του προγράμματος θα πρέπει να εκτελέσετε εντολή προς εντολή για πολλή ώρα μέχρι να το βρείτε.

Για να αποφύγετε αυτή τη ταλαιπωρία, μπορείτε να χρησιμοποιήσετε breakpoints - ειδικά σημάδια του προσομοιωτή που σταματούν την εκτέλεση όταν έρθει η σειρά της εντολής όπου βάλατε το breakpoint. Πριν προσθέσετε ένα breakpoint, πρέπει να κάνετε assemble τον κώδικά σας. Αν δεν υπάρχουν λάθη, στο execute tab, παράθυρο Text Segment, που περιέχει το πρόγραμμά σας, θα δείτε αριστερά από κάθε εντολή ένα check-box. Σε όποια εντολή θέλετε να βάλετε breakpoint κάντε κλικ στο check-box ώστε να είναι σημαδεμένο. Από εδώ και πέρα λίγο πρίν εκτελεστεί αυτή η εντολή, ο MARS θα σταματάει και θα μπορείτε να δείτε τις τιμές καταχωρητών και μνήμης. Επίσης θα μπορείτε να συνεχίσετε την εκτέλεση, είτε εντολή προς εντολή ώστε να βρείτε το λάθος, είτε μέχρι το επόμενο breakpoint.

4 Παραδοτέο

Το μόνο παραδοτέο είναι το αρχείο lab02.asm. Στον κατάλογο test θα βρείτε το `lab02_tester.py` που εξετάζει 4 περιπτώσεις που το πρόγραμμά σας θα πρέπει να αντιμετωπίζει με επιτυχία. Δεν χρειάζεται να αλλάξετε κάποια αναμενόμενη τιμή ή κάτι άλλο σε αυτό το αρχείο για την άσκηση.

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

Τα προγράμματά σας θα βαθμολογηθούν για την ορθότητά τους, την ποιότητα σχολίων και τη ταχύτητα εκτέλεσής τους. Το τελευταίο σημαίνει ότι πρέπει να είναι σύντομα και ο αριθμός εντολών, ειδικά μέσα στον βρόγχο, να είναι όσο γίνεται μικρότερος.