2ο Εργαστήριο Αρχιτεκτονικής Η/Υ: MIPS assembly: Παλίνδρομα

Α. Ευθυμίου

Παραδοτέο: Παρασκευή 6 Μαρτίου 2015, 20:00

Το αντικείμενο αυτής της άσκησης είναι ένα πρόγραμμα που ελέγχει αν μία λέξη/πρόταση είναι παλίνδρομο. Σε αυτή την εργαστηριακή άσκηση θα γράψετε προγράμματα assembly που χρησιμοποιούν εντολές σύγκρισης και διακλάδοσης. Θα πρέπει να έχετε μελετήσει τα μαθήματα για τη γλώσσα assembly του MIPS που αντιστοιχούν μέχρι και την ενότητα 2.7 του βιβλίου.

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

1 Παλίνδρομα

Τα παλίνδρομα είναι λέξεις που είναι ίδιες αν τις διαβάσει κανείς είτε από δεξιά προς τα αριστερά είτε από αριστερά προς τα δεξιά. Ενα παράδειγμα είναι η λέξη racecar.

Κατ'επέκταση υπάρχουν και παλίνδρομες προτάσεις, μόνο που σ'αυτή τη περίπτωση αγνοούμε τη διαφορά πεζών-κεφαλαίων, τα σημεία στίξης και τα κενά. Για παράδειγμα: No lemons no melon.

Τα παλίνδρομα δεν έχουν καμιά ιδιαίτερα σπουδαία εφαρμογή. Έχουν όμως πλάκα. Υπάρχει ένα τραγούδι από τον weird Al Yankovic, με τίτλο Bob όπου όλοι οι στίχοι είναι παλίνδρομα, όπως βέβαια και ο τίτλος του. Θα το βρείτε εδώ. Παρεμπιπτόντως ο Yankovic έχει γράψει τον εθνικό ύμνο των αστικών λεωφορείων και ένα τραγούδι σχετικό με το αντικείμενο του μαθήματός μας!

2 Περιγραφή του αλγορίθμου

Για να ελέγξει κανείς αν ένα string είναι παλίνδρομο, ξεκινάει εξετάζοντας τα δύο ακριανά γράμματα: το πρώτο και το τελευταίο. Αν είναι ίδια συνεχίζει με τα επόμενα, μέχρι είτε να βρεθούν δύο διαφορετικά γράμματα, είτε να φτάσουμε στο μέσο του string. Η συνθήκη `φτάσαμε στο μέσο' θέλει λίγη προσοχή γιατί μπορεί το string να περιέχει άρτιο ή περιττό αριθμό χαρακτήρων.

Για την άσκηση θα θεωρήσουμε ότι τα strings είναι κωδικοποιημένα ως ASCII, περιέχουν μόνο πεζά γράμματα και ότι τελειώνουν με τον ειδικό χαρακτήρα `'\0'`, που έχει τιμή 0.

3 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 την αρχή του προγράμματος.

4 Πρώτη Υλοποίηση

Ο κώδικας που σας δίνεται βρίσκει τον τελευταίο χαρακτήρα του string και αποθηκεύει τη διεύθυνσή του στον καταχωρητή `$s2`. Η διεύθυνση του πρώτου χαρακτήρα είναι στον καταχωρητή `$s1`.

Εσείς θα πρέπει να συνεχίσετε με την υλοποίηση του αλγορίθμου που εξετάζει αν το string είναι παλίνδρομο. **Αν το string είναι παλίνδρομο πρέπει, όταν καλείτε το syscall εξόδου (με** `$v0=10`$_{ten}$**), ο καταχωρητής** `$a0` **να έχει την τιμή 0. Αλλιώς θα πρέπει να έχει την τιμή 1.** Προσοχή οι τιμές του `$a0`

είναι ανάποδες από ότι μπορεί να περιμένατε! Ακολουθούν τη σύμβαση του Unix όπου το 0 σημαίνει «όλα καλά» ενώ άλλες τιμές δείχνουν ότι έγινε κάποιο λάθος.

Δε χρειάζεται να ελέγχετε ότι το string δεν περιέχει άλλους χαρακτήρες ή ότι δεν είναι άδειο.

5 Δεύτερη Υλοποίηση - Bonus

Βελτιώστε την προηγούμενη υλοποίηση ώστε να μπορεί να χειριστεί και strings που περιέχουν κενά. Τα κενά θα πρέπει να αγνοούνται. Δεν χρειάζεται να ελέγχετε ότι το string περιέχει μόνο κενά ή άλλους χαρακτήρες (π.χ. σημεία στίξης κλπ).

Κρατήστε το αρχείο με την πρώτη υλοποίηση ως έχει και σώστε τον κώδικά σας με όνομα lab02v2.asm, για να μπορώ να δώ ότι κάνατε τη δεύτερη υλοποίηση.

Αν δουλεύει σωστά η δεύτερη υλοποίηση, ο βαθμός της άσκησης θα πολλαπλασιαστεί με 1.15. Φυσικά αν πάει πάνω από 10 θα βοηθήσει τον μέσο όρο των ασκήσεων, αλλά ο συνολικός βαθμός ασκήσεων δεν μπορεί να περάσει το 30/100.

6 MARS Debugging

Τώρα που τα προγράμματά σας έχουν γίνει μεγαλύτερα και πιο πολύπλοκα, ο απλοϊκός τρόπος αποσφαλμάτωσης με εκτέλεση εντολή προς εντολή, δεν είναι πλέον αποδοτικός. Αν αντιμετωπίζετε ένα πρόβλημα που βρίσκεται στο τμήμα του κώδικα που υλοποιεί τον έλεγχο παλινδρόμου, θα πρέπει να εκτελέσετε εντολή προς εντολή όλη την επανάληψη που βρίσκει το τέλος του string.

Για να αποφύγετε αυτή τη ταλαιπωρία, θα πρέπει 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.

7 Παραδοτέο

Το αρχείο lab02.asm θα πρέπει να έχει την πρώτη υλοποίηση ανίχνευσης παλινδρόμου. Αν δοκιμάσατε τη δεύτερη υλοποίηση, ονομάστε το αρχείο lab02v2.asm και μη ξεχάσετε να το συμπεριλάβετε στο repository. Πρέπει να κάνετε commit τις αλλαγές σας και να τις στείλετε (push) στο GitHub repository 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 στους βρόγχους, να είναι όσο γίνεται μικρότερος.