4ο Εργαστήριο Αρχιτεκτονικής Η/Υ: Αναδρομικός έλεγχος ταξινόμησης ενός πίνακα από string

Α. Ευθυμίου

Παραδοτέο: Βλ. ecourse, 23:59

Με αυτή την εργαστηριακή άσκηση θα γράψετε ένα αναδρομικό πρόγραμμα σε Risc-V assembly που ελέγχει αν ένας πίνακας από strings είναι ταξινομημένος σε αύξουσα σειρά. Θα πρέπει να έχετε μελετήσει μέχρι και το 5ο μάθημα για τη γλώσσα assembly του Risc-V, που αναφέρεται σε υπορουτίνες στον Risc-V.

1 Παραλαβή του σκελετού της άσκησης

Ακολουθήστε τον σύνδεσμο που υπάρχει στο ecourse για να δημιουργηθει το δικό σας αποθετήριο της άσκησης και κλωνοποιήστε το.

Στον κατάλογο που θα δημιουργηθεί, `lab04-ghUsername`, θα βρείτε το αρχείο `lab04.s`, με έναν μικρό σκελετό κώδικα και και μια δοκιμαστική εικόνα για είσοδο, καθώς και το `Lab04Test.py`, για έλεγχο που περιέχει αρκετές περιπτώσεις εισόδων για έλεγχο.

2 Η άσκηση

Για την άσκηση θα υλοποιήσετε δύο υπορουτίνες, μία για να συγκρίνει δύο strings και μία που ελέγχει αν ένας πίνακας από strings είναι ταξινομημένος. Η δεύτερη θα καλεί την πρώτη για να εξετάσει την κατάταξη των string ανά δύο.

2.1 Σύγκριση strings

Η πρώτη υπορουτίνα, με όνομα, str_ge, θα δέχεται τις διευθύνσεις δύο string στους καταχωρητές `a0`, `a1` και θα επιστρέφει (στον `a0`) την τιμή 1 αν το string στη διεύθυνση `a0` είναι λεξικογραφικά μετά το string στη διεύθυνση `a1`, ή τα δύο string είναι όμοια. Αλλιώς θα επιστρέφει 0. Το "λεξικογραφικά μετά" σημαίνει ότι σε ένα λεξικό το string στο `a0` θα ήταν μετά από αυτό στο `a1`. Προσοχή όμως η κατάταξη των χαρακτήρων ακολουθεί την κωδικοποίηση ASCII.

Παραδείγματα:

- `str_ge("ABC", "ACB")` επιστρέφει 0 γιατί το πρώτο γράμμα είναι ίδιο, αλλά το `C` είναι μετά το `B`. Επομένως ολόκληρο το `ABC` είναι πριν από το `ACB`.
- `str_ge("a", "B")` επιστρέφει 1 γιατί το `a` εχει κωδικό 97 ενώ το `B` 66. Γενικά **όλα** τα κεφαλαία είναι πριν από τα πεζά γράμματα.
- `str_ge("abc", "abc.")` επιστρέφει 0 γιατί το δεύτερο string έχει ένα χαρακτήρα περισσότερο στο τέλος, την τελεία.

Η υπορουτίνα μπορεί να υλοποιηθεί είτε με επαναλήψεις, είτε αναδρομικά. Το ευκολότερο είναι να γίνει με επαναλήψεις και συνήθως αυτό δίνει καλύτερους χρόνους εκτέλεσης.

Μπορείτε να υποθέσετε ότι οι είσοδοι της υπορουτίνας είναι έγκυρες: οι διευθύνσεις των strings δεν θα είναι 0 (NULL) και τα strings τερματίζουν με ένα μηδενικό byte (`\0`). Το άδειο string, `""`, μπορεί να δωθεί ως είσοδος και παριστάνεται ως `"\0"`, δηλαδή είναι ένα byte με την τιμή 0.

2.2 Έλεγχος ταξινόμησης

Η δεύτερη υπορουτίνα, με όνομα `recCheck`, θα ελέγχει αν ένας πίνακας από strings είναι ταξινομημένος. Ως εισόδους δέχεται την διεύθυνση του πίνακα, στον `a0` και το μέγεθός του, αριθμός των string, στον `a1`. Η τιμή επιστροφής (στον `a0`) είναι η τιμή 1 αν ο πίνακας είναι ταξινομημένος και 0 αν δεν είναι.

Ένας πίνακας από strings είναι ένας πίνακας από δείκτες που αποτελούν τις διευθύνσεις μνήμης όπου ξεκινά το κάθε string. Σε C θα δηλωνόταν ως `char *table[size]`.

Η υπορουτίνα αυτή **πρέπει να είναι αναδρομική**[1] και να χρησιμοποιεί την υπορουτίνα `str_ge` για να συγκρίνει δύο string κάθε φορά.

Ο ψευτοκώδικας της υπορουτίνας δίνεται παρακάτω:

```
int recCheck(char *table[], int size)
{
  if (size == 0 or size == 1)
    return 1;
  if (str_ge(table[1], table[0]))      # if first two items in ascending order,
    return recCheck(&(table[1]), size-1);  # check from 2nd element onwards
  else
    return 0;
}
```

Μπορείτε να υποθέσετε ότι οι είσοδοι της υπορουτίνας είναι έγκυρες: το μέγεθος θα είναι θετικός αριθμός ή μηδέν, η διεύθυνση του πίνακα δεν θα είναι 0 (NULL) και τα strings θα τερματίζουν με ένα μηδενικό byte (`\0`).

3 Παραδοτέο

Το παραδοτέο της άσκησης είναι το αρχείο lab04.s που περιέχει το πρόγραμμά σας. Μην αλλάξετε το όνομα του αρχείου, γράψετε εκτός της περιοχής που δείχνουν τα σχόλια, γιατί δεν θα το βρίσκει ο αυτόματος έλεγχος! Προεραιτικά αλλάξτε και το README.md.

Πρέπει να κάνετε commit τις αλλαγές σας και να τις στείλετε (push) στο αποθετήριό σας στο GitHub.

Θα πρέπει να ακολουθήσετε την σύμβαση του Risc-V για την χρήση των καταχωρητών. Πρέπει να προσέξετε πώς περνάτε τιμές στις υπορουτίνες, τί είδους καταχωρητές να χρησιμοποιήσετε και πότε και αν χρειάζεται να αποθηκεύσετε τιμές στη στοίβα και μετά να τις ξαναδιαβάσετε. Το τεστ πραγματοποιεί κάποιους σχετικούς ελέγχους και σταματάει την εκτέλεση δίνοντας ένα μήνυμα λάθους.

[1] Μπορεί να υλοποιηθεί με επαναλήψεις ευκολότερα, αλλά χρησιμοποιώντας αναδρομή θα καταλάβετε σε βάθος πως δουλεύει ένας επεξεργαστής.