3η εργαστηριακή άσκηση

Σχεδιασμός Ενσωματωμένων Συστημάτων

Αρβανίτης Χρήστος: 03114622
Μπαγάκης Μάνος: 03114157

Στην άσκηση αυτή ζητήθηκε η υλοποίηση τριών προγραμμάτων σε assembly συμβατή με τον επεξεργαστή ARM. Στη συνέχεια σκιαγραφείται η υλοποίηση καθενός από τα προγράμματα που ζητήθηκαν. Σημειώνεται πως ο αντίστοιχος κώδικας περιλαμβάνεται στο παραδοτέο συμπιεσμένο αρχείο.

Ερώτημα 1

Για αυτό το ερώτημα, χρησιμοποιούνται οι κλήσεις scanf και printf (αντί των αντίστοιχων sys calls). Συγκεκριμένα, προκειμένου να υλοποιήσουμε τη λειτουργικότητα που απαιτείται, σε κάθε επανάληψη της συνεχούς λειτουργίας, εκτυπώνουμε το prompt:

Input a string of up to 32 char long:

Στη συνέχεια διαβάζουμε το input του χρήστη από τη scanf και το αποθηκεύουμε στη κατάλληλη θέση μνήμης, η οποία για τις επόμενες 33 θέσεις είναι αρχικοποιημένη με τον κενό χαρακτήρα '\0'.

Έχοντάς το αποθηκευμένο ελέγχουμε χρησιμοποιώντας την *comparStrings* για ισότητα με τα `q` και Q ώστε αν αυτή αληθεύει να τερματίζουμε το πρόγραμμα με κατάλληλη έξοδο.

Έπειτα τυπώνουμε το prompt εξόδου:

Result is:

και καλούμε την *transform_string*, η οποία και διεξάγει τους ελέγχους βάσει των ascii τιμών του string εισόδου.
Πιο συγκεκριμένα με διαδοχικούς ελέγχους αποφασίζουμε για τη μετατροπή που χρειάζεται το σύμβολο που διαβάσαμε και διακρίνουμε τις εξής περιπτώσεις:

1. Το σύμβολο δεν είναι λατινικό γράμμα ή αριθμός : Δεν υφίσταται κάποια αλλαγή

2. Το σύμβολο είναι λατινικό πεζό : Το σύμβολο μετατρέπεται στο αντίστοιχο κεφαλαίο λατινικό.
3. Το σύμβολο είναι λατινικό κεφαλαίο: Το σύμβολο μετατρέπεται στο αντίστοιχο πεζό λατινικό.
4. Το σύμβολο είναι αριθμός: Ο αριθμός αυξάνεται κατά 5 αν είναι μικρότερος του 5, αλλιώς μειώνεται κατά 5.

Το αποτέλεσμα κάθε μετατροπής αποθηκεύεται στην αντίστοιχη θέση του inp_str_tran ενώ παράλληλα καθαρίζεται ο buffer που χρησιμοποιήθηκε για το διάβασμα του χρήστη ώστε να είναι έτοιμος για την επόμενη επανάληψη του προγράμματος.

Ο αντίστοιχος κώδικας περιλαμβάνεται στο αρχείο που υποβάλλεται

Ερώτημα 2

Σε αυτή την άσκηση επιλέξαμε να χρησιμοποιήσουμε system calls για κάθε ενδεχόμενο I/O. Για αυτό το ερώτημα, χρησιμοποιούνται οι κλήσεις συστήματος read και write στους κατάλληλους δείκτες αρχείων (stdin & stdout). Συγκεκριμένα, προκειμένου να υλοποιήσουμε τη λειτουργικότητα που απαιτείται, σε κάθε επανάληψη της συνεχούς λειτουργίας, εκτυπώνουμε το prompt:

Input a string of up to 32 char long:

Στη συνέχεια διαβάζουμε το input του χρήστη από τη read και το αποθηκεύουμε στη κατάλληλη θέση μνήμης, η οποία για τις επόμενες 33 θέσεις είναι αρχικοποιημένη με τον κενό χαρακτήρα ‘\0’.

Στην άσκηση απαιτείται να αγνοούνται οι χαρακτήρες πλέον των 32 αρχικών (ή καλύτερα των 33 με το \n της εισόδου). Για αυτόν τον λόγο, υλοποιούμε μια αντίστοιχη της fflush της C συνάρτηση. Συγκεκριμένα, αν ο 33ος χαρακτήρας δεν είναι ‘\n’ τότε διαβάζουμε απο το stdin μέχρι να μην υπάρχει άλλος χαρακτήρας, αποθηκεύοντας το αποτέλεσμα σε έναν dump buffer.

Έχοντάς το αποθηκευμένο ελέγχουμε χρησιμοποιώντας την *comparStrings* για ισότητα με τα `q` και Q ώστε αν αυτή αληθεύει να τερματίζουμε το πρόγραμμα με κατάλληλη έξοδο.

Επιπλέον αυτών, αρχικοποιούμε έναν πίνακα ώστε να φυλλάξει την συχνότητα εμφάνισης κάθε πιθανού ascii χαρακτήρα εκτός των πρώτων 33.

Έπειτα καλούμε την *transform_string*, η οποία και απαρτίζεται από δύο μέρη. Αυτό της μέτρησης της συχνότητας των χαρακτήρων και αυτό της εκτύπωσης στο αρχείο.
Σχετικά με το πρώτο μέρος, διαβάζουμε κάθε χαρακτήρα της εισόδου και αφαιρούμε 32 ώστε να έχουμε τον πρώτο αποδεκτό χαρακτήρα στην πρώτη θέση του πίνακα. Εδώ πρέπει να

σημειωθεί πως ελέγχουμε πριν ώστε να μην δεχόμαστε τους προηγούμενους χαρακτήρες, αφού κάτι τέτοιο θα οδηγούσε σε εξαιρετικά μεγάλο index του πίνακα οπότε και segmentation fault.

Αφού καταμετρήσουμε όλους τους χαρακτήρες περνάμε στο σκέλος της εκτύπωσης στο αρχείο. Μια εκτύπωση σε αρχείο απλά απαιτεί ως file pointer τη διεύθυνση του αρχείου αυτού. Προς αυτή τη κατεύθυνση, χρησιμοποιούμε την open κληση συστήματος οπότε και αποκτάμε τον file pointer.

Ας σταθούμε στην open. Επιλέξαμε σε αυτή την υλοποίηση να ανοίγουμε το αρχείο output.txt με τα δικαιώματα O_WRONLY | O_CREAT | O_APPEND προκειμένου αν υπάρχει ήδη το αρχείο να τοποθετούμε στο τέλος του τα νέα δεδομένα.

Η εκτύπωση των αποτελεσμάτων πρέπει να γίνεται με τη σειρά εμφάνισης του κάθε γράμματος. Για αυτόν τον λόγο διαπερνάμε ξανά κάθε έναν από τους χαρακτήρες του input και, αφού αφαιρέσουμε τη κατάλληλη τιμή προκειμένου να κάνουμε index, λαμβάνουμε το πλήθος εμφανίσεων. Φροντίζουμε, αφού λάβουμε το πλήθος να μηδενίσουμε την αντίστοιχη θέση του frequency πίνακα ώστε να είναι έτοιμος για την επόμενη επανάληψη και να μην ξανατυπώσουμε το αποτέλεσμα σε επόμενη εμφάνιση.

Η εκτύπωση των αποτελεσμάτων έχει την μορφή:
{char} -> {cnt}
συνεπώς φροντίζουμε να εκτυπώνουμε κάθε φορά και το σήμα του βέλους στο αρχείο. Πρέπει να σημειωθεί πως λαμβάνουμε ως αριθμό το πλήθος εμφανίσεων και οφείλουμε να το μετατρέψουμε σε συμβολοσειρά. Για αυτόν τον λόγο εφαρμόζουμε για κάθε ψηφίο την γνωστή από προηγούμενα μαθήματα μετατροπή δεκαδικού αριθμού σε συμβολοσειρά. Το πρόγραμμα μεταβαίνει στην επόμενη επανάληψη.

Στη συνέχεια παραθέτουμε ένα παράδειγμα εκτέλεσης για πέντε διαφορετικές φράσεις
Input a string of up to 32 char long: Hello world!

Input a string of up to 32 char long: i SHOT THE SHERIFF

Input a string of up to 32 char long: i hate embedded systems

Input a string of up to 32 char long: baby got back

Input a string of up to 32 char long: i like ma coffee like ma women

Input a string of up to 32 char long: q

Output.txt
H -> 1
e -> 1

l -> 3
o -> 2
w -> 1
r -> 1
d -> 1
! -> 1

i -> 1
S -> 2
H -> 3
O -> 1
T -> 2
E -> 2
R -> 1
I -> 1
F -> 2

i -> 1
h -> 1
a -> 1
t -> 2
e -> 5
m -> 2
b -> 1
d -> 3
s -> 3
y -> 1

b -> 3
a -> 2
y -> 1
g -> 1
o -> 1
t -> 1
c -> 1
k -> 1

i -> 3
l -> 2
k -> 2
e -> 5
m -> 3
a -> 2

c -> 1
o -> 2
f -> 2
w -> 1
n -> 1

Ερώτημα 3

Σε αυτό το ερώτημα υλοποιούμε συναρτήσεις σε assembly, οι οποίες οφείλουν να προσφέρουν την ίδια διεπαφή με τις αντίστοιχες της string.h. Οι συναρτήσεις αυτές είναι οι ακόλουθες:

- **strcmp()**: Η συνάρτηση αυτή δέχεται δύο συμβολοσειρές στους r0,r1. Συγκρίνουμε ένα ένα τους χαρακτήρες των συμβολοσειρών μεταξύ τους. Αν είναι ίδιοι οι χαρακτήρες συνεχίζουμε στον επόμενο ενώ αν είναι διαφορετικοί επιστρέφουμε -1 ή 1(αν ο χαρακτήρας του r0 είναι μικρότερος ή μεγαλύτερος αντίστοιχα). Επιστρέφει 0 αν είναι οι συμβολοσειρές ίσες.
- **strlen()**: Η συνάρτηση αυτή προσπελαύνει την συμβολοσειρά στον r0 έως ότου δεχθεί έναν null character. Επιστρέφει την τιμή του μετρητή κατά την προσπέλαση.
- **strcpy()**: Η συνάρτηση αυτή δέχεται δύο συμβολοσειρές στους r0,r1. Στην συνέχεια γράφει κάθε χαρακτήρα του r1 στον r0 (overwrite) έως ότου διαβάσει null character, ο οποίος θα είναι και ο τελευταίος που θα εγγραφεί στην r0.
- **strcat()**: Η συνάρτηση αυτή δέχεται ως όρισμα δύο συμβολοσειρές στους r0,r1. Ξεκινώντας από το τελευταίο χαρακτήρα του r0 (μέσω της strlen), γράφουμε τους χαρακτήρες του r1 στον r0, ενώνοντας έτσι τις δύο συμβολοσειρές.

Αφού υλοποιήσουμε τις παραπάνω συναρτήσεις αλλάζουμε, όπως αναφέρεται στην εκφώνηση της άσκησης, την αρχή του main αρχείου δοκιμής που δίνεται. Συγκεκριμένα αφαιρούμε το #include <string.h> και προσθέτουμε τα εξής:

```
extern int strcmp(char * a, char * b);
extern int strlen(char * a);
extern char* strcpy(char * a,char* b);
extern int strcat(char * a,char* b);
```