ΣΧΕΔΙΑΣΜΟΣ ΕΝΣΩΜΑΤΩΜΕΝΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΧΕΙΜΕΡΙΝΟ ΕΞΑΜΗΝΟ, 2017-2018

Άσκηση 2η : Εργασία σε assembly του επεξεργαστή ARM

Ερώτημα 1ο: Μετατροπή εισόδου από τερματικό

Να γραφεί πρόγραμμα **σε assembly του επεξεργαστή ARM** το οποίο θα λαμβάνει ως είσοδο από τον χρήστη μέσω τερματικού, μια συμβολοσειρά μεγέθους έως 32 χαρακτήρων. Αν η είσοδος είναι μεγαλύτερη, οι χαρακτήρες που περισσεύουν, να αγνοούνται. Στην συμβολοσειρά αυτή να γίνεται η παρακάτω μετατροπή:

- Αν ο χαρακτήρας είναι κεφαλαίο γράμμα της αγγλικής αλφαβήτου τότε να μετατρέπεται σε πεζό και αντίστροφα.

- Αν ο χαρακτήρας βρίσκεται στο εύρος [‘0’, ‘9’], θα πραγματοποιείται η ακόλουθη μετατροπή:

‘0’ → ‘5’
‘1’ → ‘6’
‘2’ → ‘7’
‘3’ → ‘8’
‘4’ → ‘9’
‘5’ → ‘0’
‘6’ → ‘1’
‘7’ → ‘2’
‘8’ → ‘3’
‘9’ → ‘4’

- Οι υπόλοιποι χαρακτήρες να παραμένουν αμετάβλητοι

Το πρόγραμμα να είναι συνεχούς λειτουργίας και να τερματίζει όταν λάβει ως είσοδο μια συμβολοσειρά μήκους ένα που θα αποτελείται από τον χαρακτήρα ‘Q’ ή ‘q’. Ο κώδικας που αναφέρεται στην μετατροπή να γραφεί ως ξεχωριστή συνάρτηση. Στόχος είναι η ελαχιστοποίηση των γραμμών κώδικα που απαιτούνται για το σώμα της εν λόγω συνάρτησης. Η υλοποίηση του προγράμματος να γίνει **με χρήση συναρτήσεων της βιβλιοθήκης της C** (scanf, printf, ….). Ένα πιθανό παράδειγμα εκτέλεσης του προγράμματος θα μπορούσε να είναι το παρακάτω:

```
$ ./ask2.1.out
Input a string of up to 32 chars long: ##asd123$X8B5mnjL9!
Result is: ##ASD678$x3b0MNJl4!

Input a string of up to 32 chars long: Q
Exiting….
$
```

Ερώτημα 2ο: Μετατροπή εισόδου από τερματικό σε αρχείο

Να γραφεί πρόγραμμα **σε assembly του επεξεργαστή ARM** το οποίο θα λαμβάνει ως είσοδο από τον χρήστη μέσω τερματικού, μια συμβολοσειρά μεγέθους έως 32 χαρακτήρων. Αν η είσοδος είναι μεγαλύτερη, οι χαρακτήρες που περισσεύουν, να αγνοούνται. Το πρόγραμμα θα επεξεργάζεται την είσοδο και θα καταμετρά πόσες φορές εμφανίζεται κάθε ascii χαρακτήρας σε αυτή. Οι πρώτοι 33 χαρακτήρες του ascii πίνακα, δηλαδή οι ειδικοί και ο κενός χαρακτήρας, να αγνοούνται. Η εμφάνιση ενός γράμματος σε κεφαλαία και μικρή μορφή να θεωρηθεί ως εμφάνιση δύο διαφορετικών χαρακτήρων. Το αποτέλεσμα να εκτυπώνεται σε αρχείο με όνομα “count.txt”, με την μορφή 'χαρακτήρας -> αριθμός εμφανίσεων', με ένα χαρακτήρα ανά γραμμή και ταξινομημένο με σειρά εμφάνισης των χαρακτήρων.

Το πρόγραμμα να είναι συνεχούς λειτουργίας και να τερματίζει όταν λάβει ως είσοδο μια συμβολοσειρά μήκους ένα που θα αποτελείται από τον χαρακτήρα ‘Q’ ή ‘q’. Διαδοχικές έξοδοι, να γράφονται διαδοχικά στο αρχείο χωρισμένες από μια κενή γραμμή. Ο κώδικας που αναφέρεται στην μετατροπή να γραφεί ως ξεχωριστή συνάρτηση. Στόχος είναι η ελαχιστοποίηση των γραμμών κώδικα που απαιτούνται για το σώμα της εν λόγω συνάρτησης. Το διάβασμα της εισόδου μπορεί να υλοποιηθεί είτε με χρήση συναρτήσεων της βιβλιοθήκης της C (scanf, printf, ….) είτε με system calls του Linux (read, write, ….) είτε με συνδυασμούς και των δύο. Το γράψιμο του αρχείου εξόδου **πρέπει να πραγματοποιηθεί με χρήση system calls του Linux (open, write)**. Στην αναφορά να συμπεριληφθούν και 5 δοκιμαστικές φράσεις ανά ομάδα, καθώς και η αντίστοιχη έξοδος τους στο αρχείο.

Ένα πιθανό παράδειγμα εκτέλεσης του προγράμματος θα μπορούσε να είναι το παρακάτω:

```
$ ./ask2.2.out
Input a string of up to 32 chars long: Hello there! Nice to meet you!

Output in count.txt:
H -> 1
e -> 6
l -> 2
o -> 3
t -> 3
h -> 1
r -> 1
! -> 2
N -> 1
i -> 1
c -> 1
m -> 1
y -> 1
u -> 1

Input a string of up to 32 chars long: Q
Exiting….
$
```

Ερώτημα 3ο: Σύνδεση κώδικα C με κώδικα assembly του επεξεργαστή ARM.

Σκοπός της παρούσας άσκησης είναι να συνδυαστεί κώδικας γραμμένος σε C με συναρτήσεις γραμμένες σε assembly του επεξεργαστή ARM. Στον σύνδεσμο Έγγραφα→Ασκηση_2→ Άσκηση βρίσκεται ένα αρχείο που ονομάζεται string_manipulation.c. Το πρόγραμμα αυτό, ανοίγει ένα αρχείο με 512 γραμμές, κάθε γραμμή του οποίου περιέχει μια τυχαία κατασκευασμένη συμβολοσειρά, μεγέθους από 8 έως 64 χαρακτήρες. Κατά την εκτέλεση του προγράμματος κατασκευάζονται 3 αρχεία εξόδου:

1. Το πρώτο περιέχει το μήκος της κάθε γραμμής του αρχείου εισόδου.
2. Το δεύτερο περιέχει τις συμβολοσειρές του αρχείου εισόδου ενωμένες (concatenated) ανά 2.
3. Το τρίτο περιέχει τις συμβολοσειρές του αρχείου εισόδου ταξινομημένες σε αύξουσα αλφαβητική σειρά.

Για να επιτευχθούν οι παραπάνω στόχοι γίνεται χρήση των συναρτήσεων **strlen**, **strcpy**, **strcat** και **strcmp** από την βιβλιοθήκη string.h. Στόχος σας είναι να αντικαταστήσετε τις παραπάνω συναρτήσεις με δικές σας γραμμένες σε ARM assembly.

Για την σύνδεση assembly και γλώσσας C υπάρχουν δυο διαθέσιμοι τρόποι:

i. Να γραφούν συναρτήσεις τις C, που θα περιέχουν inline assembly εντολές κάνοντας χρήση της εντολής asm. Για παράδειγμα:

```
asm("MOV r0, #5");
__asm__("MOV r1, #10");
```

Όπως βλέπουμε μπορούμε να χρησιμοποιήσουμε τόσο την συνάρτηση asm() όσο και την __asm__(), σε περίπτωση που έχουμε δηλώσει κάποια μεταβλητή ως asm.

ii. Οι συναρτήσεις που θέλουμε να υλοποιήσουμε, δηλώνονται ως extern στον κώδικα της C. Έστω ότι έχουμε ένα αρχείο με πηγαίο κώδικα σε C το οποίο ονομάζεται my_prog.c και περιέχει μια συνάρτηση foo δηλωμένη ως extern. Ο πηγαίος κώδικας της συνάρτησης foo γράφεται σε ένα άλλο αρχείο που περιέχει κώδικα assembly και έστω ότι ονομάζεται my_fun.s. Απαραίτητο είναι η foo να έχει δηλωθεί στο κώδικα assembly με το directive .global για να μπορεί να είναι ορατή από τον linker κατά την σύνδεση των δυο αρχείων. Για να παραχθεί το τελικό εκτελέσιμο αρχείο, ακολουθούμε τα παρακάτω βήματα:

- Κάνουμε μόνο compile (-c flag του gcc) το αρχείο my_prog.c.

```
$ gcc –Wall –g my_prog.c –c my_prog
```

- Κάνουμε μόνο compile το αρχείο my_fun.s

```
$ gcc –Wall –g my_fun.s –c my_fun
```

- Συνδέουμε (link) τα object αρχεία που έχουν παραχθεί από τα παραπάνω βήματα, για την παραγωγή του τελικού εκτελέσιμου αρχείου.

Στα πλαίσια της άσκησης θα χρησιμοποιήσουμε αυστηρά τον τρόπο 2. Συνοψίζοντας, δεν χρειάζεται να αλλάξετε κάτι στον κώδικα C που συνοδεύει την άσκηση, πέρα από το να δηλώσετε τις συναρτήσεις ως extern και να σβήσετε την εντολή #include <string.h>. Οι δηλώσεις των συναρτήσεων **strlen**, **strcpy**, **strcat** και **strcmp** πρέπει να είναι σε μορφή ακριβώς ίδια με αυτή που προδιαγράφεται όταν εκτελέσετε στο τερματικό σας την εντολή man string.

Παραδοτέος κώδικας της άσκησης θα είναι το αρχείο ή αρχεία που περιέχουν τον πηγαίο κώδικα των συναρτήσεων σας σε assembly και ένα makefile για την σωστή μεταγλώττιση και σύνδεση των επιμέρους αρχείων. Το παραγόμενο εκτελέσιμο πρέπει να έχει όνομα **string_manipulation.out**. Στον φάκελο της άσκησης υπάρχουν επίσης δυο example input αρχεία με ονόματα rand_str_input_first και rand_str_input_sec.

Υποσημείωση: Ο gcc εξ ορισμού παράγει όταν μπορεί κώδικα για το Thumb instruction set ώστε να μειώσει το μέγεθος του παραγόμενου εκτελέσιμου αρχείου. Για τον λόγο αυτό προτείνεται να ξεκινάτε την παραγόμενη συνάρτηση σας με τουλάχιστον τα παρακάτω directives:

```
.text
.align  4          /* allignment του κώδικα
.global foo        /* όνομα συνάρτησης */
.type   foo, %function
```

Φυσικά αν θέλετε και εσείς να χρησιμοποιήσετε το Thumb instruction set, είστε ελεύθεροι .Προσοχή όμως στα directives προς τον assembler που θα χρησιμοποιήσετε.

Γενικές παρατηρήσεις:

1. Οι κώδικες σας, να συνοδεύονται από έναν στοιχειώδη σχολιασμό στα βασικά τους σημεία.
2. Θα ληφθεί υπόψη η σωστή χρήση του instruction set του ARM, για παραγωγή κώδικα μειωμένου μεγέθους. Η μη χρήση αυτών των χαρακτηριστικών θεωρείται μη αποδοτική επίλυση και δεν θα βαθμολογηθεί με τον μέγιστο βαθμό.
3. Οι κώδικες πρέπει να συνοδεύονται από μια αναφορά που θα συνοψίζει σε λίγες γραμμές το σκεπτικό πίσω από την υλοποίηση του κάθε προγράμματος σας.
4. Η παράδοση της άσκησης θα γίνει είτε ατομικά είτε σε ομάδες των δύο ατόμων.
5. **Το ανανεωμένο περιβάλλον qemu, επιτρέπει την χρήση του GNU Debugger (gdb) τόσο για την γλώσσα C όσο και για την assembly. Η χρήση του θα διευκολύνει πάρα πολύ την αποσφαλμάτωση όλων των ασκήσεων. Προσοχή στα flags του gcc που χρειάζονται για την σωστή χρήση του gdb. Πέρα από τον gdb διαθέσιμα είναι και άλλα βοηθητικά εργαλεία για προγραμματισμό συστήματος όπως π.χ το strace.**