ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ & ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Σχεδιασμός Ενσωματωμένων Συστημάτων
9ο Εξαμηνο ΗΜΜΥ

Συνεργάτες: Γεώργιος-Ταξιάρχης Γιαννιός, Α.Μ.: 03116156
Μπέτζελος Χρήστος, Α.Μ.: 03116067

3η Εργαστηριακή Άσκηση:
ΕΡΓΑΣΙΑ ΣΕ ASSEMBLY ΤΟΥ ΕΠΕΞΕΡΓΑΣΤΗ ARM

Ερώτημα 1ο : Μετατροπή εισόδου από τερματικό

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

- Αν ο χαρακτήρας είναι κεφαλαίο γράμμα της αγγλικής αλφαβήτου τότε μετατρέπεται σε πεζό και αντίστροφα.
- Αν ο χαρακτήρας βρίσκεται στο εύρος [‘0’, ‘9’], 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 παραμένουν αμετάβλητοι.

Το πρόγραμμα είναι συνεχούς λειτουργίας και τερματίζει όταν λάβει ως είσοδο μια συμβολοσειρά μήκους ένα που θα αποτελείται από τον χαρακτήρα ‘Q’ ή ‘q’. Ο κώδικας που αναφέρεται στην μετατροπή γράφτηκε ως ξεχωριστή συνάρτηση. Στόχος μας ήταν η ελαχιστοποίηση των γραμμών κώδικα που απαιτούνται για το σώμα της εν λόγω συνάρτησης. Η υλοποίηση του προγράμματος έγινε με χρήση system calls του Linux (read, write, exit). Η μεταγλώττιση και το τρέξιμο των προγραμμάτων έγινε σύμφωνα με τις οδηγίες που μας δόθηκαν με το ανανεωμένο περιβάλλον qemu και την χρήση του GNU Debugger (gdb). Για την σωστή χρήση του gdb τρέχουμε gcc με flag -g στο αρχείο ask1.s που υπάρχει και στο zip.

Τα στάδια που ακολουθούνται στο εν λόγω πρόγραμμα είναι τα εξής:

- Γράφει μέσω syscall write “Input a string of up to 32 chars long: ”.

- Διαβάζει μέσω syscall read την είσοδο του χρήστη.

- Ελέγχει αν το μήκος της εισόδου είναι 2 (ένας χαρακτήρας και το EOL). Αν ναι, ελέγχει αν ο χαρακτήρας είναι q ή Q. Αν είναι, καλεί το syscall exit για να τερματίσει. Αν όχι, (και στις δύο περιπτώσεις) περνάει στην συνάρτηση *convert string*.

Η συνάρτηση *convert string* δέχεται ως όρισμα στον r3 το μήκος της συμβολοσειράς. Η συνάρτηση διατρέχει τους χαρακτήρες που υπάρχουν στην θέση μνήμης του r1, μέχρι να είναι το r3 ίσο με 0. Όσο το κάνει αυτό, φορτώνει το στοιχείο που είναι στην θέση μνήμης r1 στον r0. Η συνάρτηση έχει τα εξής βήματα:

- Ελέγχει αν είναι μεγαλύτερο του ASCII χαρακτήρα 9. Αν είναι, τότε υποθέτει ότι είναι γράμμα. Αν δεν είναι, ελέγχει αν είναι μεγαλύτερο από τον χαρακτήρα 0. Αν είναι, τότε σίγουρα είναι αριθμός. Αν δεν είναι τότε, δεν τον αλλάζουμε αυτό τον χαρακτήρα και πηγαίνουμε στον επόμενο.

- Αν ο χαρακτήρας είναι αριθμός, ο νέος αριθμός που θέλουμε είναι ο αρχικός επαυξημένος κάτα 5 και mod 10. Αφαιρεί τον χαρακτήρα 0 σε ASCII και κάνει την πρόσθεση με 5. Αν ο αριθμός είναι μεγαλύτερος από 10, τότε αφαιρούμε το 10 για να φτάσουμε στο επιθυμητό αποτέλεσμα. Τέλος προσθέτουμε τον χαρακτήρα 0 ξανά για να επανακατασκευάσουμε τον χαρακτήρα μας.

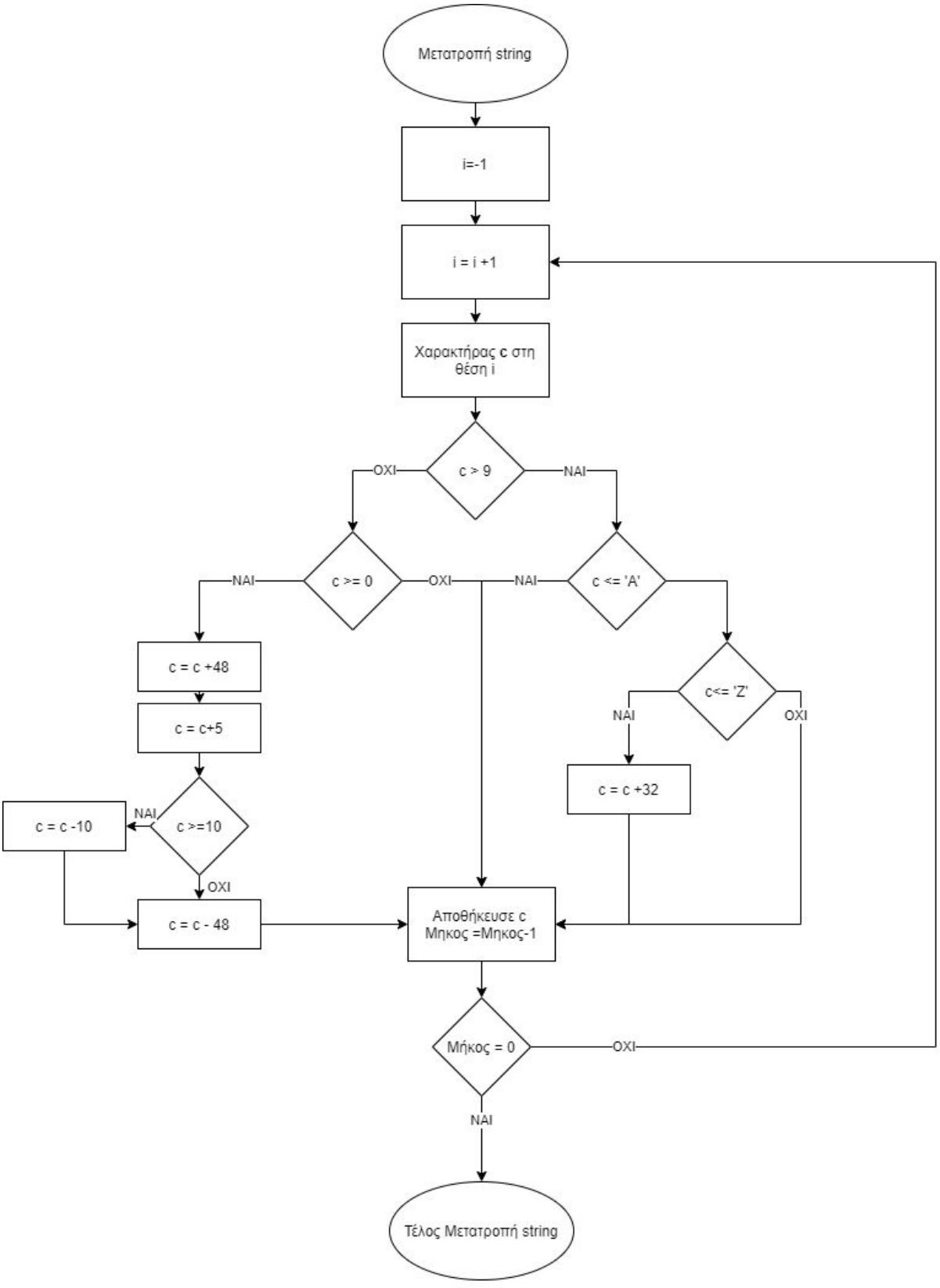
- Αν ο χαρακτήρας είναι μεταξύ 65 και 90, τότε είναι κεφαλαίο γράμμα και προσθέτουμε 32 για να το κάνουμε μικρό.

- Αν ο χαρακτήρας είναι μεταξύ 97 και 122, τότε είναι μικρό γράμμα, οπότε αφαιρούμε 32 για να το κάνουμε κεφαλαίο.

Τέλος ελέγχουμε αν η συμβολοσειρά είναι ακριβώς 32 χαρακτήρες. Τότε, αν ο τελευταίος χαρακτήρας δεν είναι EOL, τον προσθέτουμε στο τέλος.

Επίσης, πρέπει να καθαρίσουμε τους υπόλοιπους χαρακτήρες (αν έχουμε γράψει παραπάνω από 32) από το stdin. Αυτό το κάνουμε πάλι με read(). Αν στο read πάρουμε λιγότερους από 32 χαρακτήρες, τότε είμαστε καλά και μεταβαίνουμε πάλι στην αρχή για να επαναληφθεί η διαδικασία. Αλλιώς ελέγχουμε την οριακή περίπτωση που είναι ακριβώς 32 χαρακτήρες αυτοί που απομένουν. Σε αυτή την περίπτωση ο τελευταίος χαρακτήρας είναι EOL, οπότε επιστρέφουμε στην αρχή του προγράμματος, αλλιώς διαβάζουμε και άλλο.

Το **διάγραμμα ροής** της συνάρτησης *convert string* φαίνεται παρακάτω.


Μετατροπή string
i=-1
i = i +1
Χαρακτήρας c στη θέση i
c > 9
ΟΧΙ
ΝΑΙ
c >= 0
ΝΑΙ
ΟΧΙ
ΝΑΙ
c <= 'A'
c = c +48
c = c+5
c >=10
ΝΑΙ
c = c -10
ΟΧΙ
c = c - 48
c<= 'Z'
ΝΑΙ
ΟΧΙ
c = c +32
Αποθήκευσε c
Μηκος =Μηκος-1
Μήκος = 0
ΟΧΙ
ΝΑΙ
Τέλος Μετατροπή string

Ερώτημα 2ο : Επικοινωνία guest και host μηχανημάτων μέσω σειριακής θύρας

Δημιουργήθηκαν 2 προγράμματα, ένα σε C στο **host** μηχάνημα και ένα σε assembly του ARM στο **guest** μηχάνημα τα οποία επικοινωνούν μέσω εικονικής σειριακής θύρας. Το πρόγραμμα στο host μηχάνημα δέχεται ως είσοδο έναν string μεγέθους έως 64 χαρακτήρων. Το string αυτό αποστέλλεται μέσω σειριακής θύρας στο guest μηχάνημα και αυτό με την σειρά του απαντάει ποιος είναι ο χαρακτήρας του string με την μεγαλύτερη συχνότητα εμφάνισης και πόσες φορές εμφανίστηκε. Ο κενός χαρακτήρας (Ascii no 32) εξαιρείται από την καταμέτρηση. Στην περίπτωση που δύο ή παραπάνω χαρακτήρες έχουν μέγιστη συχνότητα εμφάνισης, το πρόγραμμα επιστρέφει στον host, τον χαρακτήρα με τον μικρότερο ascii κωδικό.

Για την επικοινωνία και την ρύθμιση του διαύλου, χρησιμοποιήσαμε την βιβλιοθήκη *termios*. Στο guest μηχάνημα, καλέσαμε την εξωτερική συνάρτηση *tcsetattr* για να ρυθμίσουμε τα απαραίτητα options.

Η επιλογή των τιμών για τις σημαίες έγινε μετά από δοκιμές, έχοντας τρέξει κώδικα c στο guest μηχάνημα. Οι έξτρα εντολές που τρέξαμε στο πρόγραμμα c από την πλευρά του guest για την παραμετροποίηση φαίνονται με σχόλια στον κώδικα host.c.

```
root@debian-armel:~# gcc host.c -o host
root@debian-armel:~# ./host
Please give a string to send to guest:
bbbaaaaaaa
iflag: 0, oflag: 4, lflag: a22 cflag: 8bd
```

Έχοντας βρει τις τιμές για τις σημαίες, τις τοποθετήσαμε στα options του τελικού assembly κώδικα guest.s και έτσι πετύχαμε ο host και ο guest να έχουν κοινό configuration σειριακής θύρας. Αφού παραμετροποιήσαμε τη σειριακή θύρα, προχωρήσαμε στην υλοποίηση των επιμέρους απαιτήσεων:

- Ο host διαβάζει από το stdin την συμβολοσειρά που του γράφει ο χρήστης από το τερματικό.
- Ανοίγει τη θύρα και στέλνει την συμβολοσειρά στον guest, ο οποίος προφανώς ήδη έχει ανοίξει τη θύρα και περιμένει σε read να διαβάσει.

- Ο guest διαβάζει τη συμβολοσειρά, την αποθηκεύει σε θέσεις μνήμης, την διατρέχει και αυξάνει σε ένα πίνακα συχνοτήτων τις τιμές εμφάνισης του κάθε χαρακτήρα που βρίσκει. Αυτός ο πίνακας έχει 256 entries, όσοι και οι βασικοί ascii κωδικοί και είναι αρχικοποιημένος με μηδέν.
- Μετά διατρέχει τον πίνακα συχνοτήτων και κρατάει το index του πίνακα (το οποίο είναι το ascii code του χαρακτήρα) και την τιμή του. (Ο πίνακας διατρέχεται από το μηδέν μέχρι και το 256, αγνοώντας τον κενό χαρακτήρα 32. Με αυτόν τον τρόπο εξασφαλίζεται και ότι σε περίπτωση ισοπαλίας, επιστρέφεται ο χαρακτήρας με τον μικρότερο ascii κωδικό.)
- Στη συνέχεια, γράφει αυτές τις δύο τιμές (char, freq) στη θύρα, ώστε να τις διαβάσει ο host και να εκτυπώσει το αποτέλεσμα στο stdout.

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

Το πρόγραμμα *string_manipulation.c*, ανοίγει ένα αρχείο εισόδου 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*. Αντικαταστήσαμε τις παραπάνω συναρτήσεις με δικές μας γραμμένες σε ARM assembly.

Από τους δύο τρόπους που μας παρουσιάστηκαν για την σύνδεση assembly και γλώσσας C, επιλέξαμε και χρησιμοποιήσαμε αυστηρά τον δεύτερο τρόπο. Συγκεκριμένα, αρχικά, για κάθε μία συνάρτηση, την δηλώσαμε ως extern στον κώδικα C και ο πηγαίος κώδικάς της γράφτηκε σε ένα άλλο αρχείο που περιέχει κώδικα assembly. Απαραίτητο ήταν η συνάρτηση να έχει δηλωθεί στο κώδικα assembly με το directive *.global* για να μπορεί να είναι ορατή από τον linker κατά την σύνδεση των δύο αρχείων. Στη συνέχεια, κάναμε μόνο compile (*-c flag του gcc*) το αρχείο *string_manipulation.c* και το ίδιο για κάθε ένα αρχείο .s των συναρτήσεων. Συνδέσαμε (link) τα object αρχεία που παρήχθησαν από τα παραπάνω βήματα, για την παραγωγή του τελικού εκτελέσιμου αρχείου.

Δεν χρειάστηκε να αλλάξουμε κάτι στον κώδικα C, πέρα από το να δηλώσουμε τις συναρτήσεις ως extern και να σβήσουμε την εντολή #include . Οι δηλώσεις των συναρτήσεων *strlen*, *strcpy*, *strcat* και *strcmp* είναι σε μορφή ακριβώς ίδια με αυτή που προδιαγράφεται όταν εκτελούμε στο τερματικό την εντολή *man string*. Συγκεκριμένα έχουμε τις εξής υλοποιήσεις:

strlen

- Αρχικοποιούμε τον counter στο μηδέν.
- Διατρέχουμε το string μας και όσο δεν βλέπουμε τον null χαρακτήρα, αυξάνουμε τον counter κατά ένα.
- Επιστρέφουμε τον counter.

strcpy

- Σώζουμε τη διεύθυνση βάσης του dst string.
- Διατρέχουμε το src string μας και όσο δεν βλέπουμε τον null χαρακτήρα, αποθηκεύουμε τον χαρακτήρα στο dst string μας.
- Επιστρέφουμε τη διεύθυνση βάσης του dst string.

strcat

- Σώζουμε τη διεύθυνση βάσης του dst string.
- Διατρέχουμε το dst string μας μέχρις ότου να διαβάσουμε τον null χαρακτήρα. Έτσι, αποθηκεύουμε την address του τέλους του dst string.
- Διατρέχουμε το src string μας και όσο δεν βλέπουμε τον null χαρακτήρα, αποθηκεύουμε τον χαρακτήρα στο τέλος του dst string μας.
- Επιστρέφουμε τη διεύθυνση βάσης του dst string.

strcmp

- Αρχικοποιούμε το αποτέλεσμα στο μηδέν.
- Διατρέχουμε παράλληλα και τα δύο strings. Όσο διαβάζουμε ίσους χαρακτήρες προχωράμε στον επόμενο.
- Αν το char του string 1 είναι μεγαλύτερο του char του string 2, τότε επιστρέφουμε ένα.
- Αν ισχύει το αντίστροφο, επιστρέφουμε μείον ένα.
- Αν φτάσουμε στο τέλος και των δύο strings σημαίνει ότι είναι ίσα, οπότε επιστρέφουμε μηδέν.

Στο zip αρχείο υπάρχει και ένα *makefile* για την σωστή μεταγλώττιση και σύνδεση των επιμέρους αρχείων. Το παραγόμενο εκτελέσιμο έχει όνομα *string_manipulation.out*. Χρησιμοποιήσαμε τα δύο example input αρχεία με ονόματα *rand_str_input_first* και *rand_str_input_sec* και τα αποτελέσματα φαίνονται στο φάκελο *results*.