ΑΡΙΣΤΟΤΕΛΕΙΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΟΝΙΚΗΣ

Τμημα ΗΜΜΥ. Τομεας Ηλεκτρονικης

Μικροεπεξεργαστες και Περιφερειακα

1η Εργασία

Συντάκτης :
Χρηστος Χουτουριδης
ΑΕΜ:8997
cchoutou@ece.auth.gr

Διδάσκον :
Παπαετσταθιου Ιωαννης
ygp@ece.auth.gr

3 Μαΐου 2020

1. Εισαγωγη

Στην παρούσα εργασία το ζητούμενο είναι η υλοποίηση μιας ρουτίνας σε γλώσσα assemblly ARM, που να ελέγχει αν ένα αλφαριθμητικό είναι παλίνδρομο, αλλά και η ενσωμάτωση αυτής σε ένα πρόγραμμα σε γλώσσα C. Για την εργασία χρησιμοποιήσαμε το εργαλείο Keil uVision και μιας και δεν αντιμετωπίσαμε κάποιο ιδιαίτερο πρόβλημα, η αναφορά αυτή θα αναλωθεί κυρίως στην υλοποίηση και τον έλεγχο του προγράμματος. Τον κώδικα της εργασίας αλλά και της παρούσας αναφοράς μπορείτε να βρείτε ακόμα στο προσωπικό αποθετήριο της εργασίας.

2. Παραδοτεα

Στο παραδοτέο .zip αρχείο μπορείτε να βρείτε:

- Το αρχείο **main.c** που περιέχει όλο τον ζητούμενο κώδικα της εργασίας.
- Το αρχείο **report.pdf** που είναι παρούσα αναφορά.
- Τον κατάλογο **Keil** που περιέχει το project στο Keil που χρησιμοποιήσαμε.
  Στο project αυτό περιέχονται επιπλέων οι ρυθμίσεις καθώς και τα αρχεία από τη βιβλιοθήκη CMSIS που χρειάστηκαν για την ορθή αρχικοποίηση και αλληλεπίδραση με τον επεξεργαστή.

3. Υλοποιηση

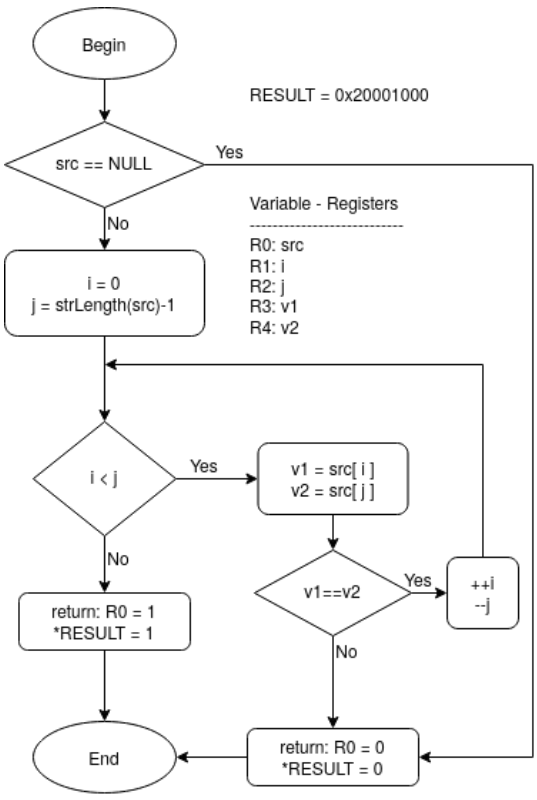
Η υλοποίηση του ελέγχου αν ένα αλφαριθμητικό είναι παλίνδρομο έγινε σε δύο συναρτήσεις, την *isPalindrome()* και την *strLength()*, που λειτουργεί ως βοηθητική συνάρτηση. Οι συναρτήσεις αυτές είναι υλοποιημένες σε γλώσσα assemblly αλλά η κλήση τους από το κυρίως πρόγραμμα δεν διαφέρει καθόλου από την κλήση μιας οποιασδήποτε άλλης συνάρτησης σε γλώσσα C. Για τον λόγο αυτό δεν θα ασχοληθούμε περαιτέρω με το κύριο πρόγραμμα (συνάρτηση main).

3.1. Συνάρτηση *isPalindrome()*

Για τη συνάρτηση αυτή χρησιμοποιήσαμε ένα απλό O(n) σειριακό αλγόριθμο, ο οποίος απαιτεί το μήκος του αλφαριθμητικού εισόδου προτού εισέλθει στον κύριο βρόχο επανάληψης. Για να βρούμε το μήκος χρησιμοποιούμε μια ξεχωριστή συνάρτηση. Αυτή την υλοποιήσαμε από την αρχή ώστε να μην χρειαστεί να χρησιμοποιηθεί η συνάρτηση *strlen()* της *libc*.

Όπως φαίνεται και στο διάγραμμα του σχήματος 1, στον κύριο βρόχο της συνάρτησης χρησιμοποιούμε δύο απαριθμητές i, j χρησιμοποιώντας τους καταχωρητές R1, R2 αντίστοιχα και ένα δείκτη στη διεύθυνση του αλφαριθμητικού χρησιμοποιώντας τον καταχωρητή R0. Ο ένας δείκτης εκκινεί από την αρχή του αλφαριθμητικού και ο άλλος από το τέλος. Σε κάθε επανάληψη διαβάζουμε από το αλφαριθμητικό τα src[i] και src[j] και ελέγχουμε αν είναι ίσα. Αν δεν είναι τότε το αλφαριθμητικό δεν είναι παλίνδρομο. Αν είναι τότε μεταθέτουμε το i μία θέση δεξιά και το j μία θέση αριστερά και επανεκτελούμε τον βρόχο. Ο βρόχος συνεχίζει όσο το i είναι μικρότερο από το j. Αν το i γίνει ίσο ή μεγαλύτερο από το j, τότε ο βρόχος τερματίζει. Σε αυτή την περίπτωση το αλφαριθμητικό είναι παλίνδρομο.

Αν το αλφαριθμητικό έχει μηδενικό μήκος, τότε ο βρόχος δεν θα εκτελεστεί ούτε μία φορά και η συνάρτηση θα επιστρέψει ότι το αλφαριθμητικό είναι παλίνδρομο.



Σχήμα 1: *Διάγραμμα ροής της συνάρτησης isPalindrome()*

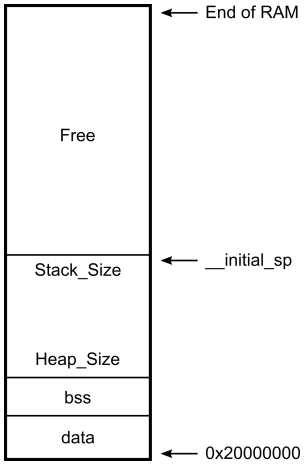
Έχουμε κάνει δηλαδή την θεώρηση ότι ***τα μηδενικού μήκους αλφαριθμητικά είναι παλίνδρομα***, καθώς είναι τα ίδια είτε τα κοιτάς από μπροστά είτε τα κοιτάς από πίσω. Στην περίπτωση που η είσοδος της συνάρτησης είναι εκτός πεδίου ορισμού, δηλαδή ο NULL pointer, τότε η συνάρτηση επιστρέφει ψευδές αποτέλεσμα. Ο έλεγχος του δείκτη εισόδου γίνεται στην αρχή της συνάρτησης πριν καν γίνει η κλήση για το μήκος, προστατεύοντας έτσι κάποιο exception από το dereference του δείκτη στην εντολή LDR.

3.2. Συνάρτηση *strLength()*

Η υλοποίηση της *strLength()* είναι επίσης ένας σειριακός αλγόριθμος. Εδώ χρησιμοποιούμε τον καταχωρητή R1 ώς δείκτη στο αλφαριθμητικό και με αυτόν κάνουμε ανάγνωση της μνήμης, αυξάνοντάς τον ταυτόχρονα μετά από κάθε ανάγνωση. Η συνάρτηση επιστρέφει τον αριθμό των αναγνώσεων που χρειάστηκαν μέχρι να λάβουμε το νούμερο ′0′, το οποίο σηματοδοτεί το τέλος του αλφαριθμητικού.

Ομοίως και εδώ κάνουμε έλεγχο της εισόδου ώστε αυτή να μην είναι ο NULL pointer. Φυσικά αυτό στη δική μας περίπτωση δεν είναι απαραίτητο καθώς υπάρχει ο αντίστοιχος έλεγχος στην *isPalindrome()* προτού την κλήση. Παρόλα αυτά το θεωρούμε καλή πρακτική, καθώς αυτό καθιστά την *strLength()* πιο ολοκληρωμένη συνάρτηση και μια χρήση της στο μέλλον δεν θα προκαλέσει εκπλήξεις.

3.3. Εγγραφή του αποτελέσματος στη μνήμη



Σχήμα 2: *Διάταξη της μνήμης*

Η συνάρτηση *isPalindrome()* αφού εκτελεστεί, επιστρέφει μια τιμή στον caller. Στα ζητούμενα όμως της εργασίας ήταν η συνάρτηση αυτή να γράφει το αποτέλεσμα και σε μία θέση μνήμης της επιλογής μας. Αυτό υπό κανονικές συνθήκες είναι ανεπίτρεπτο. Για τα πλαίσια όμως της εργασίας προσπαθήσαμε να ανταπεξέλθουμε με ένα σχετικά ασφαλή τρόπο.

Όπως φαίνεται και στο σχήμα 2, ο συνδέτης (linker) του εργαλείου keil δεν τοποθετεί τον SP στο τέλος της μνήμης. Για την ακρίβεια, χρησιμοποιώντας δύο τιμές από το configuration του startup αρχείου για το μέγεθος του σωρού και της στοίβας, υπολογίζει τη τιμή ***__initial_sp***. Αυτή την τιμή χρησιμοποιεί έπειτα σαν πρώτο όρισμα στο vector table και άρα αυτή την τιμή έχει ο SP κατά την εκκίνηση.

Έχοντας αυτό σαν δεδομένο μπορούμε να επιλέξουμε μια τιμή στην ελεύθερη περιοχή, έχοντας πάντα στο μυαλό μας ότι αλλάζοντας τα δεδομένα του προγράμματος η τιμή αυτή μπορεί να αλλάξει. Στη δική μας περίπτωση η επιλεγμένη τιμή ήταν το **0x20001000**. Φυσικά αυτή η τιμή μπορεί να αλλάξει εύκολα αλλάζοντας απλώς την τιμή από το **#define RESULT**. Μάλιστα αυτό είναι και κάτι που προτρέπουμε κάνει ο οποιονδήποτε θέλει να τρέξει τον κώδικά μας, ώστε να βεβαιωθεί πως η τιμή αυτή είναι στην ασφαλή ζώνη.

4. Ελεγχος

Για τον έλεγχο της ορθότητας του κώδικα χρησιμοποιήσαμε τον debuger του εργαλείου κάνοντας ταυτόχρονη χρήση του simulator. Καθώς η εκφώνηση ανέφερε την δημιουργία μιας main, θεωρήσαμε ότι είναι ευκαιρία να χρησιμοποιήσουμε την main για να επιτύχουμε ένα είδους debug-mode unit testing. Έτσι στο εσωτερικό της δηλώσαμε αλφαριθμητικά για έλεγχο και για αυτά καλέσαμε την *isPalindrome()*. Έπειτα εκτελώντας τον debuger καταφέραμε να κάνουμε την όποια αποσφαλμάτωση που χρειάστηκε.

5. Επιλογος

Θεωρούμε ότι η παρούσα εργασία ήταν μια καλή πρώτη επαφή με την μίξη κώδικα C - assemblly, καθώς και των όποιων απαιτήσεων από μεριάς assemblly ώστε αυτή να είναι συμβατή με το πρότυπο που ακολουθεί ο compiler (AAPCS). Η μόνη παραφωνία σε αυτή τη συνεργασία ήταν η επιλογή μιας ελεύθερης θέσης μνήμης για την επιστροφή του αποτελέσματος. Αυτό γιατί μέχρι τη σύνταξη του παρόντος δεν βρήκαμε

τρόπο να διαβάσουμε από διαφορετικό αρχείο(πχ το main.c) την τιμή ___initial_sp, ώστε η επιλογή της θέσης να αυτοματοποιηθεί.