Μικροεπεξεργαστές και Περιφερειακά

1ο Εργαστήριο

Ηλιάνα Κόγια 10090
Χριστίνα Κουτσού 9994

1 Επεξήγηση υλοποίησης

1.1 main

Στη συνάρτηση main του προγράμματος ο χρήστης καλείται να δώσει ως είσοδο ένα αλφαριθμητικό της επιλογής του κατά το runtime. Καθώς το τελικό string δεν εμφανίζεται στην οθόνη, η είσοδος από το χρήστη τυπώνεται στην κονσόλα προκειμένου να διευκολύνει την επαλήθευση της εξόδου του προγράμματος που κληθήκαμε να υλοποιήσουμε. Στην περίπτωση που η τιμή που επιστρέφει η hash1 είναι αρνητική, δίνουμε σαν input στη hash2 την απόλυτη τιμή (ώστε να μην υπάρχει πρόβλημα με την εύρεση του παραγοντικού). Τέλος, η έξοδος του προγράμματος επιστρέφεται με χρήση του πρωτοκόλλου UART.

1.2 hash1

Η πρώτη ρουτίνα Assembly δέχεται ως είσοδο από τη main το αλφαριθμητικό, υπόλογίζει τη τιμή του hash και την επιστρέφει στη main. Το hash key προκύπτει προσθέτοντας για κάθε κεφαλαίο λατινικό γράμμα την αντίστοιχη τιμή από τον πίνακα hash table και αφαιρώντας κάθε ψηφίο από 1 έως 9 που υπάρχει στο string. Η τιμή του hash δεν λαμβάνει υπόψιν άλλους χαρακτήρες εκτός των 'A'-'Z', '1'-'9'. Συγκεκριμένα:
loop: Σε κάθε επανάληψη στον καταχωρητή R1 φορτώνουμε έναν χαρακτήρα του string. Συγκρίνουμε την τιμή του R1 με το 0 στο ASCII ώστε να δούμε αν είμαστε στο τέλος του string, καθώς το προαναφερόμενο είναι NULL-terminated. Στην περίπτωση της ισότητας, πραγματοποιείται branch στην exit, όπου αποθηκεύουμε την τιμή του hash στον R0(return register) και κάνουμε return στη main.

```
cmp R1, #65
blt not_letter
cmp R1, #90
bgt loop
```

Αν δεν είμαστε στο τέλος του string, τότε συγκρίνουμε τον χαρακτήρα, που είναι αποθηκευμένος στον R1, με την τιμή 65 (= 'A' in ASCII table) και αν έχει τιμή μικρότερη του 65 τότε δεν θα αντιστοιχεί σε κάποιο κεφαλαίο λατανικό γράμμα, οπότε γίνεται branch στη not_letter. Έπειτα, συγκρίνουμε την ASCII τιμή του χαρακτήρα με το 90 (= 'Z' in ASCII table) και αν είναι μεγαλύτερη από το 90 τότε ο χαρακτήρας δεν θα αντιστοιχεί σε κάποιο κεφαλαίο λατινικό γράμμα, αλλά ούτε και σε κάποιο ψηφίο μεταξύ 1-9 ('1' - '9' τιμές ASCII: 49 - 57, και 90>57). Οπότε σε αυτή την περίπτωση πραγματοποιείται branch στη loop, ώστε να μεταβούμε στον επόμενο χαρακτήρα.

```
//if letter
sub R4, R1, #65
ldrb R5, [R3,
    R4]
add R2, R2, R5
b loop
```

Αν κάποια από τις παραπάνω branch δεν εκτελεστεί, τότε ο χαρακτήρας θα είναι κεφαλαίο λατινικό γράμμα και έτσι συνεχίζουμε στον υπολογισμό του hash. Βρίσκουμε πρώτα σε ποια θέση i (καταχωρητής R4) του πίνακα hash_table βρίσκεται το key, αφαιρώντας το 65 (offset) από την τιμή του R1. Έπειτα, βρίσκουμε μέσω της ldrb το key στον hash_table και το προσθέτουμε στο hash (hash = hash + key).

not_letter: Είναι η περίπτωση που ο τρέχον χαρακτήρας δεν ανηκεί σε κάποιο από τα γράμματα 'A'-'Z' και ταυτόχρονα έχει ASCII τιμή μικρότερη του 65. Ελέγχουμε μήπως αντιστοιχεί σε ψηφίο '1' - '9', συγκρίνοντας την τιμή του χαρακτήρα με το 49 (= '1') και αν είναι μικρότερη τότε μεταβαίνουμε στον επόμενο χαρακτηρα (b loop). Αντίστοιχα αν η τιμή του χαρακτήρα εί-

ναι μεγαλύτερη από το 57 (= '9'). Αν κάποια από τις παραπάνω branch δεν εκτελεστεί, τότε ο χαρακτήρας θα είναι ψηφίο. Αφαιρούμε το 48 από την ASCII τιμή, ώστε να βρούμε το ψηφίο στο δεκαδικό, το οποίο και αφαιρούμε από το hash.

1.3 hash2

Η υπορουτίνα hash2 είναι υπεύθυνη για τον υπολογισμό του αθροίσματος των ψηφίων (έως ότου φτάσουμε σε μονοψήφιο αριθμό) του hash key και του παραγοντικού του τελευταίου.

```
sdiv R4, R0, R2 //R4 = q
mul R3, R4, R2 // R3 = q * 10
sub R5, R0, R3 // R5 =
    remainder = digit
add R1, R1, R5 // total = total
    + digit
mov R0, R4 // R0 = q
```

Στην περίπτωση που το hash key (αποθηκευμένο στον R0, ως argument της συνάρτησης) δεν είναι ίσο με το 0, κατά την πρώτη επανάληψη ο αριθμός διαιρείται με το 10 μέσω της sdiv που επιστρέφει το πηλίκο της διαίρεσης. Μέσω του πηλίκου, υπολογίζουμε το υπόλοιπο της διαίρεσης (υ = Δ - π*δ) που είναι και το τελικό ψηφίο(LSB).

Θέτοντας τώρα το πηλίκο της διαίρεσης στον R0, επανακαλείται η λούπα ώστε να καθοριστεί το επόμενο ψηφίο (LSB + 1) και να προστεθεί στο προήγουμενο. Η διαδικασία επαναλαμβάνεται μέχρι το πηλίκο της διαίρεσης να είναι ίσο με το 0. Σε περίπτωση που το άθροισμα δεν είναι μονοψήφιο, η παραπάνω διαδικασία πρέπει να πραγματοποιηθεί ξανά (bgt hash2).

Μετά τον υπολογισμό του μονοψήφιου αριθμού, ελέγχεται η περίπτωση που είναι ίσο με 0 και που οδηγεί στην επιστροφή της τιμής 1 (= 0!), αν ισχύει η ισότητα. Αλλιώς, ξεκινάει ένας βρόχος όπου η αρχική τιμή (το single sum) πολλαπλασιάζεται με έναν καταχωρητή (αποθηκεύει το factorial και αρχικοποιείται στο 1), μειώνεται κατά 1 και στη συνέχεια αντικαθιστάται. Ο βρόχος ξαναεκτελείται για τη νέα τιμή μέχρι αυτή να ισούται με τη μονάδα. Τέλος, αποθηκεύουμε το αποτέλεσμα στον καταχωρητή R0 και επιστρέφουμε στη διεύθυνση του προγράμματος όπου έγινε branch η συγκεκριμένη υπορουτίνα.

2 Testing

Προκειμένου να επιβεβαιωθεί η λειτουργικότητα των υπορουτινών Assembly χρησιμοποιήθηκε ο προσομοιωτής που διατείθεται από το Keil, χωρίς τη χρήση της UART. Μερικά παραδείγματα χρήσης είναι τα παρακάτω: