Εργαστήριο Μαθήματος: "Μικροεπεξεργαστές & Περιφερειακά"
1η Εργασία

18/4/2024

Ομάδα 18:
Μαμουγιώργη Μαρία 10533
Ξακουστού Αιμιλία 10324

main.c:

Στην main αρχικοποιήθηκε ένας μονοδιάστατος πίνακας (hash_values) που περιέχει τα στοιχεία του πίνακα της εκφώνησης. Αυτός ο πίνακας δίνεται σαν όρισμα οταν καλούμε την hash_calculation, καθώς περιέχει τις τιμές που προστίθενται ή αφαιρούνται στο hash ανάλογα με τον εκάστοτε χαρακτήρα.

```
int hash_values[26] = {10, 42, 12, 21, 7, 5, 67, 48, 69, 2, 36, 3, 19, 1, 14, 51, 71, 8, 26, 54, 75, 15, 6, 59, 13, 25}
```

hash_calculation.s:

```
extern int hash_calculation(char *str,int *hash_values)
```

Σε αυτή την συνάρτηση υπολογίζεται το hash.

Ουσιαστικά, φορτώνουμε ένα ένα τα byte του string, πραγματοποιώντας διαδοχικά για τον κάθε χαρακτήρα την ίδια διαδικασία αναγνώρισής του,(που θα εξηγήσουμε στην πορεία), ώσπου να φτάσω στον '\0' χαρακτήρα που σηματοδοτεί το τέλος του string. Από τον r0 καταχωρητή, που περιέχει την διεύθυνση του 1ου στοιχείου του string, φορτώνουμε στον r4 καταχωρητή τον χαρακτήρα μας (με χρήση της LDRB). Για να φορτώσω σταδιακά και διαδοχικά όλους τους χαρακτήρες του string,(ο κάθε χαρακτήρας δεσμεύει 1 byte), κάθε φορά που έχω τελειώσει με τον υπολογισμό του hash όσον αφορά τον χαρακτήρα μου, μετακινώ κατά 1 byte δεξιά την διεύθυνση του string (ADD r0, r0, #1) και κάνω branch στην START, όπου φορτώνεται ,πάλι από την αρχή στον r4 καταχωρητή. Το στοιχείο στο οποίο δείχνει η διεύθυνση str, η οποία προηγουμένως αυξήθηκε και δείχνει πλέον στον επόμενο χαρακτήρα.Η ανίχνευση του είδους του εκάστοτε χαρακτήρα πραγματοποιείται με διάφορες συγκρίσεις της ASCII τιμής του χαρακτήρα μας (χρήση CMP εντολής). Με αυτόν τον τρόπο ανιχνεύουμε αν είναι κεφαλαίο ή μικρό λατινικό γράμμα ή αριθμητικό ψηφίο. Αν δεν ανήκει σε κανένα από τα 3 διαστήματα, γίνεται branch στην NEXT, στην οποία πραγματοποιείται η απαιτούμενη μετακίνηση 1 byte δεξιά της διεύθυνση του string και branch πάλι στην αρχή (B START), όπου φορτώνουμε τον επόμενο χαρακτήρα του string.

Σε περίπτωση που ο χαρακτήρας είναι μικρό ή κεφαλαίο λατινικό γράμμα πραγματοποιείται η διαδικασία αντιστοίχισης της τιμής του στον πίνακα hash_values ώστε να αφαιρεθεί ή προστεθεί αντίστοιχα στην τιμή hash (r6).

Για την αντιστοίχιση των χαρακτήρων μας στις τιμές του πίνακα hash_values υπολογίζαμε αρχικά την απόστασή του από τον 1ο χαρακτήρα της κατηγορίας του (π.χ. για κεφαλαία τον

A), ώστε να βρούμε ποιος χαρακτήρας είναι στην αλφαβητική σειρά. Έπειτα για να φορτώσουμε στον r5 την τιμή του hash_values που αντιστοιχεί στον χαρακτήρα μας, πολλαπλασιάζαμε την σειρά του χαρακτήρα με το 4 (διότι οι integer μεταβλητες καταλαμβανουν 4 byte) και προσθέτοντάς το σε εναν καταχωρητη(r2), μαζί με την αρχική διεύθυνση του hash_values, ο καταχωρητής μας έδειχνε στην διεύθυνση του χαρακτήρα μας στον πίνακα hash_values.
Για να επιστρέψω το hash στην main συναρτηση, αντιγράφω το hash στον καταχωρητή r0:
MOV r0,r6

sum_function.s:

extern int sum_function(int n)

Στη συνάρτηση sum_function.s με όρισμα το hash που υπολογίστηκε προηγουμένως από την hash_calculation.s γίνεται η άθροιση κάθε ψηφίου του hash έως ότου αυτό γίνει ένας μονοψήφιος αριθμός. Στη συνέχεια αυτός ο μονοψήφιος αριθμός γίνεται το όρισμα της sum_of_natural_numbers και επιστρέφεται το άθροισμα ακεραίων του αριθμού,αναδρομικά μέχρι να γίνει ο αριθμός αυτός ίσος με μηδέν. Το άθροισμα είναι και το αποτέλεσμα που επιστρέφεται. Αρχικά όσο η τιμή του hash=r4 είναι θετική γίνεται r5=digit +=hash%10 και hash=hash/10 (όπου digit το κάθε ψηφίο του hash), ώστε να αποθηκεύεται στην τιμή digit ότι υπήρχε ήδη και το υπόλοιπο του hash, δηλαδή το πρώτο ψηφίο του. Ουσιαστικά γίνεται διάσπαση και άθροισμα των ψηφίων του hash. Στη συνέχεια ακολουθεί η ίδια διαδικασία για τη περίπτωση που το digit δεν είναι μονοψήφιος αριθμός. Όταν ολοκληρωθεί αυτή η διαδικασία , το αποτέλεσμα αυτής εκχωρείται σαν όρισμα σε μια αναδρομική συνάρτηση. Σε αυτήν για κάθε κλίση της δεσμεύεται χώρος στη στοίβα και αποδεσμεύεται στο τέλος της αναδρομής. Γίνεται ουσιαστικά ο υπολογισμός του αθροίσματος και αποθηκεύεται στον καταχωρητή r6. Στη συνέχεια γίνεται μείωση του ορίσματος r0 κατά ένα και η διαδικασία επαναλαμβάνεται μέχρις ότου το όρισμα της συνάρτησης γίνει ίσο με μηδέν,δηλαδή r0=0. Τότε, γίνεται αποδέσμευση των καταχωρητών r6,lr όπου περιέχουν αντίστοιχα το αποτέλεσμα και τη διεύθυνση της επόμενης εντολής μετά την BL, η οποία χρησιμοποιήθηκε για την επαναληπτική διαδικασία, ενός στοιχείου κάθε φορά ακολουθώντας τακτική LIFO. Έτσι μέσω της BX η εκτέλεση του προγράμματος συνεχίζει με την εντολή ADD που προσθέτει το αποτέλεσμα του αθροίσματος r6 στον καταχωρητή που περιέχει την τιμή επιστροφής r0. Η διαδικασία επαναλαμβάνεται έως ότου αποδεσμευτούν όλοι οι δεσμευμένοι καταχωρητές και ολοκληρώνεται με την επιστροφή του αθροίσματος του μονοψήφιου αριθμού που προέκυψε από τη διάσπαση και τη πρόσθεση των ψηφίων του hash r0, όπως αναλύσαμε παραπάνω.

Problems:

Τα προβλήματα με τα οποία ήρθαμε αντιμέτωπες ήταν στο κομμάτι της αναδρομικής συνάρτησης καθώς έπρεπε συνεχώς να γίνεται εισαγωγή και εξαγωγή στοιχείων στη στοίβα.

Testing:

Αρχικά υλοποιήθηκαν οι κώδικες σε γλώσσα C και επαληθεύτηκε η ομαλή και ζητούμενη λειτουργία τους. Σταδιακά μετατρέψαμε τις συναρτήσεις σε assembly και με την χρήση του debugger και του build του Keil καταλήξαμε στην τελική τους μορφή. Έγιναν πολλές δοκιμές προκειμένου να εξασφαλίσουμε τα επιθυμητά αποτελέσματα. Ένα παράδειγμα εξ αυτών είναι το αλφαριθμητικό Aa .!b42(/.FFgP 3a το οποίο δίνει hash=-49 και το αποτέλεσμα που επιστρέφεται είναι 10.