Μικροεπεξεργαστές και Περιφερειακά

Εαρινό εξάμηνο 2022-2023

Πρώτο Εργαστήριο

Δημήτριος Χαϊτίδης (ΑΕΜ: 9310)

Email: chaitidi@ece.auth.gr

Δημήτριος Κωνσταντινίδης (ΑΕΜ:10106)

Email: dakonstan@ece.auth.gr

Περιγραφή

Στο εργαστήριο αυτό καλούμαστε να υλοποιήσουμε μια ρουτίνα σε assembly ενός μικροελεγκτή ARM με την χρήση των εργαλείων Keil. Ο προγραμματισμός σε assembly περιλαμβάνει την δημιουργία του hash ενός αλφαριθμητικού. Για την υλοποίηση αυτού χρειάστηκε να δημιουργήσουμε και ένα πρόγραμμα στην C και στην συνέχεια καλούμε τις υπορουτίνες υπολογισμού του hash, που είναι γραμμένες σε assembly. Το αποτέλεσμα προβάλλεται μέσω της UART με χρήση της κατάλληλης συνάρτησης.

Η υλοποίηση των ζητουμένων γίνεται χρησιμοποιώντας 3 Script Assembly και 1 Script C, κάθε ένα για την υλοποίηση ενός ζητούμενου της εργασίας. Έτσι προκύπτουν οι παρακάτω:

- factorial.s
- ascii_conversion.s
- number_single.s
- main.c

Αρχικά στην C ορίσαμε τις απαραίτητες βιβλιοθήκες για την υλοποίηση του ζητούμενου προγράμματος όπως αυτό ορίζεται στην εργασία, δηλαδή της UART. Εκεί διαβάζουμε το αλφαριθμητικό μέσω μιας συνάρτησης scanf και ορίζουμε και τον πίνακα μετατροπής ενός χαρακτήρα στο value που δίνεται. Η UART χρησιμοποιείται για την προβολή του αποτελέσματος με την χρήση της **uart_print**,η οποία όμως επειδή δέχεται ως όρισμα ένα αλφαριθμητικό χρησιμοποιείται προηγουμένως η sprintf για την συνένωση ενός αλφαριθμητικού με το int του αποτελέσματος σε ένα νέο αλφαριθμητικό.

Ακόμη, χρειάστηκε να γίνει η σύνδεση μεταξύ της ρουτίνας C και της Assembly, η οποία γίνεται με την χρήση των :

- extern int ascii_conversion(int* result, char* str, int* character_value);
- extern int number_single(int result);
- extern int factorial(int result);

Επίσης στον κώδικα της Assembly χρειάζεται να ορίσουμε τα παρακάτω (πχ για την συνάρτηση factorial σε αυτήν την περίπτωση):

- .global factorial (Ορίζουμε ως global για να μπορεί να καλεστεί εξωτερικά)
- .p2align 2 (Γίνεται το align του location counter)
- .type factorial ,%function (Ορισμός της συνάρτησης)

Στο **ascii_conversion.s** γίνεται η μετατροπή του αλφαριθμητικού από τα ASCII values σε ένα ακέραιο με πρόσθεση και αφαίρεση των αντίστοιχων αριθμών για κάθε χαρακτήρα. Αν και αρχικά βρέθηκε η διεύθυνση των πινάκων με τον disassembler και γράφοταν/διαβάζονταν στην assembly οι κατάλληλες θέσεις μηνήμης. Με αυτήν την προσέγγιση αν δημιουργηθούν νέες μεταβλητές μπορεί να αλλα΄ξουν οι διευθύνσεις μνήμης κάθε μεταβλητής και να οδηγείται σε λανθασμένα αποτελέσματα. Έτσι η συνάρτηση δέχεται ως όρισμα 3 pointers, των 2 πινάκων και της τελικής διεύθυνσης αποθήκευσης του αποτελέσματος. Η διαδικασία μετατροπηής είναι η εξής:

- Ελέγχουμε το κάθε στοιχείο του αλφαριθμητικού εάν είναι αριθμός και κατόπιν εάν αποτελεί αποδεκτό χαρακτήρα. Σε κάθε άλλη περίπτωση το “αλφαριθμητικό” αγνοείται και προχωράμε στο επόμενο. Σε περίπτωση αριθμού, αυτός αφαιρείται από το σύνολο, σε περίπτωση χαρακτήρα, προστίθεται ο ανάλογος αριθμός σύμφωνα με τον πίνακα που μας δόθηκε. Εδώ ήρθαμε αντιμέτωποι με το πώς θα πρέπει να γίνει σωστά η αναγνώριση του αλφαριθμητικού και κατόπιν να υλοποιηθεί το ζητούμενο της εργασίας. Το αλφαριθμητικό στην assembly δίνεται με έναν pointer, που δείχνει στην αρχή της πρώτης θέσης μνήμης του πίνακα και οι χαρακτήρες αναπαριστούνται από την τιμή ASCII του κάθε χαρακτήρα. Οι χαρακτήρες (λατινικοί) σε ASCII value έχουν συνεχόμενες τιμές, οπότε αφαιρώντας την πρώτη τιμή της ακολουθίας σε ASCII (του χαρακτήρα a) λαμβάνουμε την θέση του του χαρακτήρα στον πίνακα μετατροπής. Το ίδιο γίνεται και με τους αριθμούς αφαιρώντας το ASCII value του μηδενός.
- Η διαδικασία αυτή συνεχίζεται αυξάνοντας την διεύθυνση του pointer κατά 1. Η εκτέλεση σταματαέι όταν βρεθεί τερματικός χαρακτήρας, δηλαδή το 10. Ομολογουμένως δεν ήταν ιδιαίτερα απαιτητικό.

Στο **number_single.s** υλοποιείται η διαδικασία κατά την οποία γίνεται συνεχής πρόσθεση των ψηφίων του αριθμού μέχρις ότου το αποτέλεσμα να γίνει μονοψήφιο. Για την εύρεση των διαδοχικών ψηφίων ενός αριθμού, πραγματοποιούνται διαδοχικές ακέραιες διαιρέσεις με το 10 και κάθε φορά απομονώνεται ως υπόλοιπο το τελευταίο ψηφίο. Το πηλίκο είναι τα υπόλοιπα ψηφία του αριθμού. Η υλοποίηση του συγκεκριμένου script αποτέλεσε το μεγαλύτερο μας πρόβλημα. Οι βασικές ιδέες για το παρόν ήταν τρεις.

- Η *πρώτη* είχε να κάνει με χρήση της εντολής της διάιρεσης του αριθμού με το 10 για την αφαίρεση του τελευταίου ψηφίου και την πρόσθεση του κατόπιν με το αποτέλεσμα.
- Η *δεύτερη* είχε να κάνει με την χρήση λογικής, δηλαδή με την χρήση των bits και εντολών όπως η AND για την απομόνωση των bytes και κατόπιν την πρόσθεση τους με τα υπόλοιπα ψηφία του αριθμού. Αν και λειτούργησε για κάποιες εισόδους, σε κάποιες άλλες δεν λειτούργησε.
- Η τρίτη ήταν αυτή που τελικά υλοποιήθηκε είναι αυτή των συνεχόμενων αφαιρέσεων του 10, έως ότου να μην μπορούν να πραγματοποιηθούν άλλες. Το σύνολο των αφαιρέσεων είναι το πηλίκο και το υπόλοιπο των αφαιρέσεων είναι το υπόλοιπο της διαίρεσης με το 10. Καθώς το άθροισμα των ψηφίων δεν είναι απαραίτητα μονοψήφιος αριθμός, η διαδικασία αυτή επαναλαμβάνεται μέχρι να προκύψει μονοψήφιος αριθμός.

Στο **factorial.s** γίνεται η υλοποίηση του παραγοντικού του τελικού μονοψήφιου αριθμού που προκύπτει από το number_single.s. Ο αρχικός αριθμός κρατείται σε έναν καταχωρητή, και το αποτέλεσμα

αρχικοποιείται με 1. Στην συνέχεια πολλαπλασιάζουμε το αποτέλεσμα με τον αρχικό αριθμό και έπειτα αφαιρούμε από αυτόν μια μονάδσ. Η διαδικασία επαναλαμβάνεται έως ότου φτάσουμε στο 1. Σε περίπτωση αρνητικού αριθμού το αποτέλεσμα είναι μηδέν. Επίσης τα προβλήματα του παραπάνω script δεν αποτέλεσαν ανασταλτικό παράγοντα στην υλοποίηση του, καθώς η διαδικασία είναι ιδιαίτερα απλή.

Παρατηρήσεις

Κατά την υλοποίηση της εργασίας του 1ου εργαστηρίου ήρθαμε αντιμέτωποι με ένα καινούργιο περιβάλλον εργασίας (το Keil). Εξασκήσαμε τις γνώσεις μας στην C αλλά κατά κύριο λόγω στην Assembly. Οι ρουτίνες που υλοποιηθήκαν δεν θα χαρακτηρίζονταν «δύσκολες» , ωστόσο στην προκειμένη περίπτωση η χρήση της Assembly δεν μας βοήθησε ιδιαιτέρως. Ήταν ιδιαίτερα ενδιαφέρον η πρώτη μας επαφή με τον προγραμματισμό ενός μικροελεγκτή όσο και η χρήση της UART για την επικοινωνία με αυτόν.