Εργαστήριο 1 - Ομάδα 3

Γαλανομάτης Σωκράτης - Τσακιρίδου Δήμητρα Μαρία

- Υλοποίηση κώδικα:

Υλοποιήσαμε μία συνάρτηση main με χρήση της main που περιελάμβανε το UART.zip προσθέτοντας τρεις assembly ρουτίνες και εκτυπώνοντας στο τέλος τις μεταβλητές που παράγουν.

Η πρώτη assembly ρουτίνα **compute_hash.s** παίρνει ως όρισμα το buff[] και μέσα σε μία λούπα ελέγχει από τον πρώτο μέχρι τον τελευταίο χαρακτήρα με τον εξής τρόπο: Αρχικά, ελέγχει εάν είναι μικρό λατινικό γράμμα και αν ναι, *αφαιρεί* από το sum την αντίστοιχη τιμή από το table. Εάν δεν είναι μέσα στο εύρος για να αποτελεί πεζό γράμμα κάνει branch στην ετικέτα check_upper όπου ελέγχει εάν είναι κεφαλαίο λατινικό γράμμα και *προσθέτει* στο sum την κατάλληλη τιμή από το table. Αν πάλι, ο χαρακτήρας δεν βρίσκεται μέσα σε αυτό το εύρος ελέγχει αν είναι αριθμός και αν ναι τον προσθέτει στο sum. Σε οποιαδήποτε άλλη περίπτωση ο χαρακτήρας αγνοείται. Τέλος, επιστρέφεται η τιμή του sum (r0) και αποθηκεύεται στον ακέραιο hash στην main.

Για την δεύτερη assembly ρουτίνα **compute_single_digit.s** που έπρεπε να αθροίσουμε τα ψηφία του hash μέχρι να προκύψει μονοψήφιος αριθμός, δοκιμάσαμε να χρησιμοποιήσουμε τις εντολές AND και LSR, αλλά μας άρεσε περισσότερο ο τρόπος του να διαιρούμε με το 10 με χρήση UDIV για να αφαιρέσουμε το τελευταίο ψηφίο και αφού πολλαπλασιάσουμε το παραπάνω αποτέλεσμα με το 10 με χρήση MUL, να το αφαιρούμε από το αρχικό νούμερο, κρατώντας ουσιαστικά το τελευταίο νούμερο κάθε φορά και αποθηκεύοντας το στο sum (r2). Έπειτα, βάζουμε στην θέση του αρχικού νούμερου, το νούμερο διαιρεμένο με το 10 και εκτελούμε την ίδια διαδικασία μέχρι να τελειώσουν τα ψηφία του αριθμού. Ελέγχουμε άμα το άθροισμα είναι μικρότερο του 9, εάν ναι, επιστρέφουμε το αποτέλεσμα (r0), εάν όχι, εκτελούμε την ίδια διαδικασία με το άθροισμα σαν είσοδο και μηδενίζουμε το άθροισμα για να αποθηκευτεί εκεί το νέο αποτέλεσμα.

Για την τρίτη assembly ρουτίνα **sum_of_natural_numbers.s**, αποθηκεύουμε το αποτέλεσμα της compute_single_digit.s, σε μία λούπα, προσθέτουμε το ψηφίο σε έναν καταχωρητή, το μειώνουμε κατά ένα κάθε φορά και το αποθηκεύουμε εκ νέου, μέχρι να γίνει 0.

- Προβλήματα που αντιμετωπίσαμε:

Πρακτικά προβλήματα που αντιμετωπίσαμε είναι ότι καμιά φορά μετά το debug, ο κώδικας κολλούσε (βέβαια αυτό καμιά φορά οφειλόταν σε κάποιο infinite loop που είχαμε δημιουργήσει καταλάθος). Επιπλέον, ξεκινήσαμε να δουλεύουμε με inline assembly, κάτι το οποίο μας δυσκόλεψε στο debugging γιατί δεν έπαιρνε τις εντολές της assembly κάθε μία ξεχωριστά, αλλά τις έπαιρνε σαν block εντολών και εκτελούνταν αμέσως. Τέλος, αντιμετωπίσαμε πρόβλημα στην προσπάθεια να εισάγουμε νέες βιβλιοθήκες στο ήδη υπάρχον project μας για χρήση της UART.

- Testing:

Το testing, έγινε με την χρήση του simulator, αφού κάναμε build τον κώδικα, ξεκινούσαμε το debug session όπου για να ελέγξουμε αν η υλοποίηση μας είναι σωστή, βλέπαμε πως μεταβάλλονταν οι τιμές των registers στον πίνακα αριστερά και ποιες τιμές αποθηκεύονταν στις μεταβλητές μας. Για να ελέγξουμε όσες περιπτώσεις μπορούμε, εισάγαμε διάφορα strings με πολλούς χαρακτήρες και νούμερα ώστε να δούμε ότι καλύπτεται κάθε περίπτωση.