Μικροεπεξεργαστές και Περιφερειακά
Lab 1

Ομάδα 35
Μαρινόπουλος Χριστόφορος 10522
Σπυριδόπουλος Σταύρος 10845

Εαρινό εξάμηνο 2025

Υλοποίηση Συναρτήσεων

Στο αρχείο `main.c` υλοποιήθηκε η ρουτίνα `int main(void)`.
Στο αρχείο `ASM.s` υλοποιήθηκαν 4 ρουτίνες:

- `hash`
- `addmod`
- `fib`
- `bit_xor`

Ρουτίνα 1

Στην 1η ρουτίνα υλοποιείται η `int main(void)` σε c μορφή όπου θα τρέξει το τελικό πρόγραμμα. Δηλώνονται οι συναρτήσεις που έχουν υλοποιηθεί σε assembly (`hash`, `addmod`, `fib`, `bit_xor`). Αρχικοποιείται επίσης, η τιμή του string που θα χρησιμοποιηθεί από τις asembly ρουτίνες και τυπώνεται το `hash` του.
Στη συνέχεια, αποθηκεύεται η τιμή που επιστρέφει η συνάρτηση `addmod` σε μια `int` μεταβλητή η οποία τυπώνεται και έπειτα γίνεται όρισμα στην κλήση της `fib`.
Τέλος, υπολογίζεται και τυπώνεται το bitwise XOR της αρχικής συμβολοσειράς.

Ρουτίνα 2

Η ρουτίνα `hash`, σύμφωνα με την εκφώνηση, πρέπει να δημιουργεί τον Πίνακα 1 και να τον αποθηκεύει στη μνήμη. Για την αποθήκευση του Πίνακα 1 επιλέχθηκε η θέση μνήμης `0x20001000` καθώς ανήκει στην RAM του μικροεπεξεργαστή και βρίσκεται σε ασφαλή απόσταση από τις περιοχές που χρησιμοποιούνται για global και static μεταβλητές. Μετά την δημιουργία του πίνακα, υπολογίζει το hash του ορίσματος εισόδου και το αποθηκεύει στην θέση μνήμης `0x20001100`.

Η ρουτίνα hash, υπολογίζει το μήκος του string συγκρίνοντας κάθε χαρακτήρα του με το null terminator. Όταν συναντήσει τον null termitanor έχει βρεθεί και το μήκος του string άρα και το αρχικό hash. Μετέπειτα ελέγχει κάθε χαρακτήρα ξεχωριστά βλέποντας αν είναι ο null terminator. Σε περίπτωση που είναι, αποθηκεύει το τελικό hash στη μνήμη και το επιστρέφει στην main. Στην περίπτωση που δεν είναι ο null terminator, ελέγχεται αν είναι: πρώτα αριθμός, μετά κεφαλαίο γράμμα και τέλος μικρό γράμμα. Εάν είναι γράμμα ή αριθμός κάνει τις ανάλογες πράξεις και προσθέτει το αποτέλεσμα στο αρχικό hash. Αν δεν είναι κάτι από τα παραπάνω, ελέγχει τον επόμενο χαρακτήρα.

Πίνακας 1: Πίνακας τιμών ανά ψηφίο

Ψηφίο	0	1	2	3	4	5	6	7	8	9
Τιμή	5	12	7	6	4	11	6	3	10	23

Ρουτίνα 3

Η ρουτίνα `addmod` δεν λαμβάνει ως όρισμα την τιμή του hash, αντιθέτως, την ανακτά απευθείας από τη μνήμη, συγκεκριμένα από τη διεύθυνση `0x20001100`, όπου έχει προηγουμένως αποθηκευτεί από τη ρουτίνα `hash`.
Εάν η τιμή του hash είναι μικρότερη του 10, η ρουτίνα `addmod` επιλέχθηκε να επιστρέφει την τιμή 7 (καθώς είναι ένα αποτέλεσμα που δεν μπορεί να παραχθεί από την πράξη $mod7$).
Στη συνέχεια, ώστε να υπολογιστεί το άρθοισμα των ψηφίων του hash, ακολουθείται ένας αλγόριθμος όπου πρώτα γίνεται `mod10` για να βρεθεί το δεξιότερο ψηφείο και στη συνέχεια ο αριθμός διαιρείται με το 10 ώστε να χαθεί το δεξιότερο ψηφίο. Έτσι σε κάθε επανάληψη μπορούμε να προσθέτουμε τον δεξιότερο αριθμό σε έναν καταχωρητή αθροίσματος.
Τέλος, υπολογίζει το άθροισμα των ψηφίων της, κάνει την πράξη `mod 7` και επιστρέφει το αποτέλεσμα στη main, αφού πρώτα το αποθηκεύσει στη διεύθυνση `0x20001110`.

Ρουτίνα 4

Σε αυτή τη ρουτίνα υλοποιείται ο υπολογισμός της σειράς fibonacci με αναδρομική μορφή. Αρχικά συγκρίνεται η είσοδος με τον αριθμό 1 και αν είναι μικρότερη ή ίση, επιστρέφει τον αριθμό αυτούσιο.
Αν δεν ισχύει το παραπάνω, υπολογίζεται αναδρομικά η τιμή της `fib(n-1)`. Αυτό γίνεται με την αρχική ανάθεση της τιμής $n-1$ σε έναν καταχωρητή και στη συνέχεια με την εντολή `bl fib` ξανακαλεί την ρουτίνα από την αρχή με την νέα πλέον τιμή $(n-1)$ ώς όρισμα. Η αλλιλουχία αυτή θα συνεχιστεί μέχρι $n-1=1$ όπου θα ακολουθηθεί το μονοπάτι που περιγράφηκε παραπάνω
Όταν ο υπολογισμός του $fib(1)$ φτάσει στην εντολή `bx lr`, ο link register επιστρέφει στην γραμμή `mov r5, r0` όπου αποθηκεύεται η τιμή $fib(n-1)$ στον `r5`. Έπειτα υπολογίζεται το $n-2$ και ξανακαλείται η ρουτίνα με αυτό σαν όρισμα. Ακολοθείται ξανά το αρχικό μονοπάτι για να υπολογιστεί το $fib(0)$ με τον ίδιο τρόπο.
Αφού υπολογιστεί και το $fib(0)$, η ρουτίνα φτάνει στο τέλος της και κάνει ξανά `bx lr` ώστε να επιστρέψει στον caller της. Αυτός ο caller είναι ο $fib(n-2)$, μετά ο $fib(n-1)$ και στη συνέχεις ο αρχικός $fib(n)$.
Ολόκληρη αυτή η αλληλουχία επαναλαμβάνεται μέχρι να φτάσει από το n στο 0. Περαιτέρω σχόλια και πληροφορίες για την υλοποίηση του κώδικά μπορούν να βρεθούν στο `ASM.s` αρχείο του παραδοτέου.

Ρουτίνα Bonus

Αυτή η ρουτίνα διαβάζει byte προς byte από μια συμβολοσειρά που ξεκινά στη διεύθυνση που δείχνει ο `r0`. Κάνει bitwise XOR μεταξύ όλων των χαρακτήρων μέχρι να συναντήσει το null terminator, που σηματοδοτεί το τέλος του string. Το ενδιάμεσο αποτέλεσμα αποθηκεύεται κάθε φορά στον `r2`. Μόλις ολοκληρωθεί η διαδικασία, το τελικό αποτέλεσμα αποθηκεύεται στη μνήμη στη θέση '0x20001120' και επιστρέφεται μέσω του `r0`.

Testing

Για τη διαδικασία του testing των συναρτήσεων, επιλέχθηκαν συμβολοσειρές (strings) και ακέραιοι αριθμοί (integers), για τους οποίους τα αναμενόμενα αποτελέσματα υπολογίστηκαν χειροκίνητα. Στη συνέχεια, τα αποτελέσματα αυτά συγκρίθηκαν με εκείνα που προέκυψαν από την εκτέλεση των υλοποιημένων συναρτήσεων στον κώδικα, προκειμένου να επιβεβαιωθεί η ορθότητά τους.

Προβλήματα

Δεν αντιμετωπίστηκε κάποιο πρόβλημα στην υλοποίηση των παραπάνω συναρτήσεων. Αποφάσεις σχετικές με την έλλειψη διευκρινίσεων της εκφώνησης έπρεπε να παρθούν ώστε να υπάρχει σωστή ταυτοποίηση των αποτελεσμάτων που επιστρέφονται.