Μικροεπεξεργαστές και Περιφεριακά 1

Panagiotis Koutris
10671

Christos Alexopoulos
10618

April 5, 2025

1 Κώσικας C

```c
#include <stdio.h>

int fibonacci(int n) {
    if (n == 0)
        return 0;
    else if (n == 1)
        return 1;
    else
        return fibonacci(n - 1) + fibonacci(n - 2);
}

int main() {
    char str[] = "A9b3";  // Example input string

    // === Part 1: Calculate Hash ===
    int hash = sizeof(str) - 1;  // Exclude null terminator
    int n = hash;

    int digits[10] = {5, 12, 7, 6, 4, 11, 6, 3, 10, 23};

    for (int i = 0; i < n; i++) {
        if (str[i] >= 'A' && str[i] <= 'Z') {
            hash += 2 * str[i];
        }
        else if (str[i] >= 'a' && str[i] <= 'z') {
            hash += (str[i] - 'a') * (str[i] - 'a');
        }
        else if (str[i] >= '0' && str[i] <= '9') {
            hash += digits[str[i] - '0'];
        }
```

```c
    }

    printf("Part 1 ======\n");
    printf("Hash = %d\n", hash);

    // === Part 2: Digit sum + mod7 loop ===
    int rest = 0;
    if (hash > 9) {
        int sum = 0;

        while (hash > 0) {
            sum += hash % 10;
            hash /= 10;
        }

        printf("Sum = %d\n", sum);

        while (sum > 9) {
            rest = sum % 7;
            sum /= 7;
        }
    }

    printf("Part 2 ======\n");
    printf("Remainder = %d\n", rest);

    // === Part 3: Fibonacci ===
    int fibo = fibonacci(rest);
    printf("Part 3 ======\n");
    printf("Fibonacci = %d\n", fibo);

    return 0;
}
```

Listing 1: Main C Program

2 Κώδικας assembly

Για κάθε μία από τις ρουτίνες σε assembly ακολουθήσαμε πιστά την ίδια λογική που ακολουθεί και ο κώδικας C βήμα προς βήμα και αποθηκεύσαμε τα αποτελέσματα σε global μεταβλητές στο αρχείο globals.s. Πιο αναλυτική περιγραφή της μεθοδολογίας παρατίθεται μέσω των σχολίων στον κώδικα. Δεν αντιμετωπίσαμε προβλήματα στους κώδικες και όσον αφορά το testing δοκιμάσαμε διαφορετικά παραδείγματα συμβολοσειρών για να δούμε αν βγάζουν το σωστό αποτέλεσμα για κάθε περίπτωση και πράγματι πήραμε τα επιθυμητά αποτελέσματα όπως φαίνονται και στον κώδικα της main.

3 Περιγραφή Υλοποίησης Συναρτήσεων Assembly

3.1 hash.s

Αρχικά, υπολογίζεται το μήκος της συμβολοσειράς με loop που ελέγχει για το `null terminator`. Έπειτα χρησιμοποιείται `for` loop με συγκρίσεις χαρακτήρων για να διαχωριστούν σε κατηγορίες (κεφαλαία, πεζά, ψηφία). Οι αντίστοιχοι υπολογισμοί γίνονται με εντολές όπως `ADD`, `SUB`, `MUL`, και χρησιμοποιείται πίνακας `digits[]` από integers με άμεση προσπέλαση μέσω `LSL` για μετατροπή index σε byte offset = 4 .

3.2 addmod7.s

Η `addmod7` υλοποιήθηκε με δύο διαδοχικά loops. Το πρώτο υπολογίζει το άθροισμα των ψηφίων του `hash` χρησιμοποιώντας διαίρεση και υπόλοιπο με τις εντολές `UDIV` και `MLS`. Το δεύτερο loop εφαρμόζει επαναλαμβανόμενες διαιρέσεις με το 7 όταν το άθροισμα είναι διψήφιο, μέχρι να γίνει μονοψήφιο.

3.3 fibo.s

Για τις αναδρομικές κλήσεις της ρουτίνας χρησιμοποιείται η εντολή `BL` , ενώ οι ενδιάμεσες τιμές του `r0` (το όρισμα της συνάρτησης) αποθηκεύονται προσωρινά σε καταχωρητές `r4` και `r5`. Η υλοποίηση ακολουθεί την κλασική λογική του `fibonacci(n-1) + fibonacci(n-2)`, και το τελικό αποτέλεσμα επιστρέφεται στον `r0`.