01101001
01101111
01100011
01101100
01100001

Nume și grupă:

Introducerea în Organizarea Calculatoarelor și Limbaje de Asamblare

14 ianuarie 2019

Timp de lucru: 90 de minute

1. Urmărim să facem adunarea în cruce a elementelor a doi vectori definiți în zona de date. Adică `res[i] = arr1[i] + arr2[n-i]`, unde `n` este numărul de elemente ale vectorilor. Folosiți scheletul din fișierul `cross_add.asm`.

a. Definiți doi vectori `arr1` și `arr2` în zona de date (variabile globale), care să aibă 10 elemente întregi (32 de biți). Și un vector `res` care să aibă spațiu pentru 10 elemente întregi. Scheletul afișează primul și ultimul element din cei trei vectori pentru a verifica dacă i-ați definit corespunzător. Variabila `num` din schelet reține numărul de elemente (10). (**2 puncte**)

b. Calculați în `res[0]` suma dintre primul element din `arr1` (indexul 0) și ultimul element din `arr2` (indexul `n-1`). Calculați în `res[n-1]` suma dintre ultimul element din `arr1` (indexul `n-1`) și primul element din `arr2` (indexul 0). (**4 puncte**)

c. Creați o buclă care afișează toate elementele din vectorul `arr1`, toate elementele din vectorul `arr2` și toate elementele din vectorul `res`. (**3 puncte**)

d. Înaintea buclei de mai sus creați o altă buclă care stochează în `res[i]` suma dintre `arr1[i]` și `arr2[n-i]`. (**6 puncte**)

2. Urmărim să facem un program care generează aleator un număr între 0 și 99 și apoi îl întreabă pe utilizator care este numărul și îl ajută la fiecare pas să îl ghicească indicându-i dacă numărul căutat este mai mic sau mai mare decât cel introdus. Folosiți scheletul din fișierul `guess_number.asm`.

a. Apelați funcția `rand()` din biblioteca standard C și faceți restul împărțirii la 100 și rețineți rezultatul în variabila globală `num`. (**4 puncte**)

b. Într-o buclă (infinită sau cu un număr dat de cicluri) citiți de la tastatură un număr folosind funcția `scanf()`, apoi afișați numărul. La fiecare citire și afișare stocați numărul pe stivă. Poate fi suprascris la o nouă citire. (**3 puncte**)

c. Scrieți o funcție numită `read_cmp()` care citește un număr de la tastatură, îl compară cu `num`, afișează mesajul *Corect*, *Numarul introdus este mai mic* sau *Numarul introdus este mai mare* și întoarce 1 dacă numărul a fost egal cu `num`, 0 altfel. Faceți un apel al acelei funcții. (**6 puncte**)

d. Realizați o buclă infinită în care apelați funcția `read_cmp()`. Ieșiți din buclă doar dacă rezultatul întors de funcția `read_cmp()` este 0. (**2 puncte**)

3. Urmărim să analizăm un fișier obiect și să apelăm corespunzător funcții din acel fișier. Fișierul este `hidden.o`. Dezasamblați și urmăriți conținutul fișierului `hidden.o` pentru a rezolva exercițiile. Veți scrie programul vostru în fișierul `main.c`.

Folosiți-vă de fișierul `Makefile` și de fișierul `test.c` și executabilul `test` pentru investigația funcțiilor expuse de fișierul `hidden.o`.

a. Apelați funcția `direct()` din fișierul `hidden.o` astfel încât să se afișeze mesajul *Breaking bad.* (**5 puncte**)

b. Apelați funcția `obvious()` din fișierul `hidden.o` pentru a întoarce valoarea 4242. (**5 puncte**)

c. Apelați funcția `crypto()` din fișierul `hidden.o` pentru a folosi cheia *sibling1* pentru a obține șirul decriptat *no country for old men.* (**5 puncte**)