01101001
01101111
01100011
01101100
01100001

Nume și grupă:

Introducerea în Organizarea Calculatoarelor și Limbaje de Asamblare

3 iunie 2024

Timp de lucru: 120 de minute

1. Știind că formatul binar este preferatul studenților de anul I, colegii mai mari vă testează anduranța și vă propun o serie de exerciții cel puțin interesante.

a. Dându-se două numere `a` și `b` cu `a` mai mic strict decât `b`, determinați numărul de numere pare din intervalul închis `[a, b]`, **fără a vă folosi de o operație de împărțire explicită (adică fără a vă folosi de instrucțiunea div)**. (**5 puncte**)

Hint: Operația `shr` folosită într-un anumit mod poate echivala cu împărțirea la o putere a lui 2.

b. Salvați elementele aflate pe poziții impare și care se află în intervalul închis `[a, b]` din vectorul `arr`, deja definit în program, într-un alt vector, `res`, definit de voi în orice secțiune doriți. Afișați vectorul rezultat. (**4 puncte**)

c. Afișați perechile de numere aflate pe poziții **consecutive** din `arr` care au ultimul bit diferit. (**6 puncte**)

Spre exemplu, pentru șirul de numere `10, 4, 2, 3, 5, 16`, se va afișa

```
2 3
5 16
```

Explicația este următoarea:

`2 = 10, 3 = 11`. Cele două numere au ultimul bit diferit.

`5 = 101, 16 = 10000`. Cele două numere au ultimul bit diferit.

2. Viața de student presupune pariuri fără număr, de la cele cu tine însuți, la păcănele. Unii studenți pariază că assembly-ul este inutil și este de datoria voastră să le demonstrați contrariul.

a. Creați funcția `palindrome`, cu semnătura `int palindrome(int n)` care verifică dacă un număr este palindrom. Valoarea returnată este 1, dacă numărul este palindrom sau 0, altfel. (**3 puncte**)

b. Creați funcția `read_array`, cu semnătura `void read_array(int *v, int *n)`, care citește de la tastatură un număr întreg n și apoi, n elemente ale vectorului `v`, folosind `scanf`. Apelați funcția `read_array(int *v, int *n)` cu n, o variabilă salvată pe stivă, iar v un vector declarat de voi în ce secțiune doriți. Puteți considera că numărul maxim de elemente citite este 100. (**4 puncte**)

c. Creați funcția `print_array` cu semnătura `void print_array(int n, int *v)`, care printează cele n elemente întregi ale vectorului v. Apelați funcția `void print_array(int n, int *v)` folosindu-vă de vectorul citit la punctul anterior. (**3 puncte**)

d. Creați funcția `filter_array`, cu semnătura `int filter_array(int n, int *v, int (*f)(int))`, care salvează în alt vector, definit de voi în ce secțiune doriți, elementele care produc rezultatul 1 în urma aplicării funcției f, primită ca paramatru. Funcția `filter_array` întoarce numărul de elemente salvate în noul vector. Testați implementarea folosindu-vă de vectorul citit la punctul **b.** și funcția `palindrome` implementată la punctul **a.**, împreună cu funcția de printarea de la punctul **c**. Puteți considera lungimea maximă a noului vector ca fiind 100. Astfel, dacă vectorul inițial conține valorile 12, 2, 333, 45, în urma aplicării funcției `palindrome` pe elementele sale, noul vector rezultat va conține valorile 2, 33. (**5 puncte**)

Atenție! Este interzisă apelarea directă a funcției `palindrome` pe elementele vectorului. Folosiți-vă de funcția trimisă ca parametru, pentru a păstra genericitatea!

3. Ana dorește să exploreze împreuna cu voi misterele a trei funcții scrise în limbaj de asamblare: `compute`, `check_array`, respectiv `check_string` a căror implementare se găsește în fișierul `file.asm` cu prototipurile descrise în fișierul `file.h`.

Pentru a dovedi că ați înțeles codul, deschideți fișierul `todo.c` și implementați următoarele cerințe:

a. Urmăriți conținutul funcției `todo_a`. Apelați funcția `compute` astfel încât aceasta să întoarcă rezultatul 0. Implementarea funcției `compute` se găsește în fisierul `file.asm`. Pentru verificare rulați `./main a` (**4 puncte**)

b. Urmăriți conținutul funcției `todo_b`. Inițializați corespunzător vectorul `vec` astfel încât apelul funcției `check_array` să întoarcă rezultatul 0. Pentru verificare rulați `./main b` (**5 puncte**)

c. Urmăriți conținutul funcției `todo_c`. Apelați functia `read_string` și identificați o vulnerabilitate astfel încât să se apeleze funcția `secret_function`. Pentru verificare rulați `./main c` (**6 puncte**)