01101001
01101111
01100011
01101100
01100001

Nume și grupă:

Introducerea în Organizarea Calculatoarelor și Limbaje de Asamblare

17 iunie 2024

Timp de lucru: 120 de minute

1. Pisi-Bacalhau vrea să știe dacă mâncând prea multe cod poate ajunge să se îngrașe. Ajutați-o să afle dacă a ajuns obeză.

a. Dându-se două numere pe câte 32 de biți, aflați dacă suma lor produce overflow, **fără a folosi instrucțiunea cmp**. Pentru testare, folosiți perechile de variabile `a1` și `b1` și `a2` și `b2`, deja definite în program. În cazul în care se produce overflow afișați `Go on a diet, Pisi`, altfel `You can eat more bacalhau`. **Atenție!** Este interzisă folosirea instrucțiunii `cmp`. (**5 puncte**)

b. Determinați pentru un număr pe 2 bytes primul bit setat începând din partea stângă (a reprezentării binare little endian). Pentru testare, folosiți variabila `n`, deja definită în program și afișați poziția pe care se regăsește bitul. Aceasta este o valoare din intervalul `[0, 15]`. (**5 puncte**)

c. Pentru toate elementele vectorului `arr` care au primul octet din partea dreaptă (a reprezentării little endian) egal cu 0x34, înlocuiți acest octet cu 0x61. La final, afișați vectorul rezultat ca șir de caractere. (**5 puncte**)

2. Cunoscând că securitatea argumentelor servite funcției main este importantă, studenții s-au decis să o testeze și în limbaj de asamblare.

a. În cadrul funcției `main`, citiți argumentele date în linia de comandă și salvați într-o variabilă de tip `int` reținută pe stivă, numărul de argumente care convertite, cu ajutorul funcției `atoi`, rezultă într-un număr natural nenul. Afișați contorul la final. Pentru testare, rulați `./file aaa 2 3 b 1`. (**5 puncte**)

b. În cadrul funcției `main`, parcurgeți argumentele date în linia de comandă și salvați într-un vector declarat pe stivă toate numerele naturale nenule conținute de acest șir. **Atenție!** Elementele salvate în vectorul rezultat trebuie să fie de tip int (adică vor fi convertite tot cu ajutorul funcției `atoi`). Afișați vectorul rezultat, folosindu-vă, de asemenea, de contorul determinat la punctul **a.**. (**5 puncte**)

c. Creați funcția `to_int` cu semnătura `int to_int(int n, int *v)` care returnează concatenarea celor n **cifre** reținute în vectorul dat ca argument. Spre exemplu, pentru vectorul 1, 2, 3, 4, 5, funcția `to_int` returnează numărul 12345. Este garantat faptul că rezultatul începe într-un element de tip `int` și faptul că **nu aveți de concatenat decât cifre**. Pentru verificare folosiți vectorul salvat pe stivă la punctul **b.**, și numărul de elemente calculat la punctul **a.**. **Atenție!** Pentru acest subpunct, aveți grijă să dați ca argumente executabilului numere formate dintr-o singură cifră nenulă. (**5 puncte**)

3. Mr. A a găsit o rețetă nouă pentru a face chec pufos. Vreți și voi să aflați această nouă rețetă?

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

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

c. Urmăriți conținutul funcției `todo_c`. Apelați functia `catch_me` și identificați o vulnerabilitate în fișierul obiect hidden.o încât să se apeleze funcția `secret`. Pentru verificare rulați `./main c` (**6 puncte**)