Tema 2 - Strings

- **Deadline HARD: ~~29.11.2020~~ 03.12.2020, ora 23:55**
- Data publicării: 09.11.2020
- Responsabili:
  - Cristian-Mihai VIJELIE
  - Gabriel-Iulian OLARU
- Updates: 09-11-2020, ora 22:54

Enunț

Mașina timpului lui Gigel s-a defectat. Rămânând blocat în timpul pandemiei de ciumă bubonică, el și-a folosit toate talentele pentru a găsi un leac. Victorios, Gigel realizează că este important ca formula leacului său să fie bine pazită pentru a nu fi distrusă, așa că decide să învețe diverse soluții de criptare. Reușește să salveze din rămășițele mașinii timpului un editor de text și un asamblor nasm pentru arhitectura Intel x86.

Este recomandată parcurgerea `Laboratorului 5: Rolul registrelor, adresare directă și bazată` înainte de începerea temei

Structură și detalii de implementare

Tema este formată din mai multe subpuncte, fiecare subpunct constând in implementarea unei funcții de criptare sau de prelucrare a unui șir de caractere. Subpunctele pot fi rezolvate independent, însă puteți refolosi fragmente din rezolvarea unui subpunct în rezolvarea altor subpuncte, acolo unde considerați necesar. Fiecare subpunct va fi rezolvat într-un fișier separat.

Parametrii funcțiilor sunt plasați în registre, în cadrul scheletului. Rezultatul fiecărei funcții se va plasa în primul parametru al funcției.

Scheletul include și macro-ul PRINTF32, folosit in laborator, pentru a vă ajuta la depanarea problemelor. Tema finală nu trebuie sa facă afișări folosind PRINTF32, functii externe sau apeluri de sistem

în tema finală este interzisă apelarea funcțiilor externe

1. One Time Pad - 10p

Prima oara, Gigel dorește să implementeze One Time Pad. Algoritmul constă in efectuarea operției XOR între un mesaj(plain) și o cheie(key), de aceeași lungime cu mesajul, astfel:

```
cipher[i] = plain[i] ^ key[i]
```

Se cere implementarea, în assembly, a funcției care realizează criptarea One Time Pad.

Antetul funcției, în C, este:

```
void otp(char *ciphertext, char *plaintext, char *key, int length)
```

2. Caesar Cipher - 15p

Nemulțumit de securitatea soluției precedente, Gigel încearcă și Cifrul Caesar. Acesta primește un mesaj și o cheie, reprezentată de un număr, și deplasează circular fiecare literă din mesaj cu valoarea specificată de cheie. Prin deplasare circulară se înțelege că literele mici vor rămâne, după deplasare, litere mici, iar literele mari vor rămane litere mari.

Exemplu de criptare, pentru mesajul "Azazel", folosind cheile 0, 1, 2, 3:

Mesaj	Cheie	Codificare
Azazel	0	Azazel
	1	Babafm
	2	Cbcbgn
	3	Dcdcho

Antetul funcției, în C, este:

```
void caesar(char *ciphertext, char *plaintext, int key, int length)
```

Caracterele care nu sunt litere nu vor fi criptate.

3. Vigenere Cipher - 25p

Dorind să își păstreze opțiunile deschise, Gigel incearcă și Cifrul Vigenere. Acesta primește mesajul ce urmeaza să fie criptat și o cheie, reprezentată de un șir de majuscule. În cazul în care cheia este mai scurtă decât mesajul, aceasta se extinde la lungimea mesajului, prin repetarea cheii initiale. Apoi, fiecare literă din mesaj este deplasată circular la dreapta de un număr de ori egal cu poziția în alfabet (începând de la poziția 0) a literei corespunzătoare din cheie.

Exemplu de criptare:

```
Mesaj:         Donald Trump
Cheie:         BIDEN
Cheie extinsă: BIDENB IDENBI
Mesaj criptat: Ewqeye Buyzq
```

- Anunțuri
- Bune practici
- Calendar
- Catalog
- Feed RSS
- IOCLA Need to Know
- Reguli și notare
- Resurse utile

Cursuri

- Curs 00: Prezentare
- Curs 01-02: Programe și sistemul de calcul
- Curs 02-03: Arhitectura sistemelor de calcul
- Curs 03: Arhitectura x86
- Curs 04: Reprezentarea datelor in sistemele de calcul
- Curs 05: Reprezentarea datelor in sistemele de calcul - C2
- Curs 06 - 07: Setul de instructiuni
- Curs 07: Declararea datelor
- Curs 08 - 09: Moduri de adresare
- Curs 09: Stiva
- Curs 10 - 11: Functii
- Curs 12: C + asm
- Curs 13: Unelte, utilitare
- Curs 13 - 15: Buffer overflows, securitate
- Curs 16 - 17: Optimizări
- Curs 18 - 19: Virgulă flotantă

Laboratoare

- Laborator 01: Reprezentarea numerelor, operații pe biți și lucru cu memoria
- Laborator 02: Operații cu memoria. Introducere în GDB
- Laborator 03: Toolchain
- Laborator 04: Introducere în limbajul de asamblare
- Laborator 05: Rolul registrelor, adresare directă și bazată
- Laborator 06: Lucrul cu stiva
- Laborator 07: Apeluri de funcții
- Laborator 08: Structuri, vectori. Operatii pe șiruri
- Laborator 09: Interactiunea C-assembly
- Laborator 10: Gestiunea bufferelor. Buffer overflow
- Laborator 11: Optimizări
- Laborator 12: Linking

Teme

- Tema 1 - printf
- Tema 2 - strings
- Tema 3 - AST

Table of Contents

Prima literă din mesaj, 'D', este deplasată cu o poziție la dreapta, devenind 'E', întrucât poziția literei corespunzătoare din cheia extinsă, 'B', în alfabet, este 1 (numerotarea începe de la 0).

Caracterele care nu sunt litere vor fi ignorate. Criptarea mesajului este echivalenta cu criptarea mesajului din care se pastreaza doar literele

Antetul funcției, în C, este:

```
void vigenere(char *ciphertext, char *plaintext, int plaintext_len, char *key, int key_len)
```

4. StrStr - 25p

Realizând că formula lui este destul de alambicată, Gigel dorește un mod rapid de a găsi cuvinte chieie. SrtStr este o funcție ce găsește prima apariție a unui subșir intr-un șir sursă. Se cere implementarea sa in assembley.

Antetul funcției, în C, este:

```
void my_strstr(int *substr_index, char *haystack, char *needle, int haystack_len, int needle_len);
```

În primul parametru al funcției veți retine indexul primei apariții a subșirului in șir.

Dacă subsecvența nu se regăsește in șirul original, veți returna lungimea șirului original + 1

5. Binary to Hexadecimal - 15p

Pentru a fi mai ușor de depanat problemele cu programele sale, Gigel dorește să înțeleagă puțin mai bine limbajul mașina. Scopul este realizarea unei funcții ce va trece numerele din baza 2 in baza 16.

Antetul funcției, în C, este:

```
void bin_to_hex(char *hexa_value, char *bin_sequence, int length);
```

Veti transforma secventa de biti primita ca parametru in corespondenta sa din baza 16.

```
  1011111011101111 -->    1011 1110 1110 1111    --> beef
```

Precizări suplimentare

În schelet este inclus și checker-ul, împreună cu testele folosite de acesta. Pentru a executa toate testele, se pot folosi comenzile `make run` sau `./checker`, după ce checker-ul a fost compilat, folosind `make`. Pentru a testa doar un subpunct, se poate folosi `./checker <n>` unde n este numărul subpunctului.

Pentru a putea compila checker-ul, fisierele cu cod sursa (.asm) trebuie mutate din directorul skel/ in directorul checker/ sau se poate folosi comanda `SKEL_FOLDER=../skel/ make`

Formatul fișierelor de intrare pentru One Time Pad și Caesar este:

```
length
plaintext
key
```

Formatul fișierelor de intrare pentru Vigenere este:

```
plain_len key_len
plaintext
key
```

Formatul fișierelor de intrare pentru bin_to_hex este:

```
length
binary_text
```

Formatul fișierelor de intrare pentru my_strstr este:

```
str_length substr_length
string
substring
```

Trimitere și notare

Temele vor trebui încărcate pe platforma vmchecker (în secțiunea IOCLA) și vor fi testate automat.

Arhiva încărcată va fi o arhivă `.zip` care trebuie să conțină:

- fișierele sursă ce conțin implementarea temei: `otp.asm` `caesar.asm` `vigenere.asm` `bin_to_hex.asm` `my_strstr.asm`
- `README` ce conține descrierea implementării

Punctajul final acordat pe o temă este compus din:

- punctajul obținut prin testarea automată de pe vmchecker - 90%
- README + coding style - 10%.

Se va ține cont de:

- claritatea codului
- indentare consecventa
- comentarii
- nume sugestive pentru label-uri
- fișier README

Temele care nu trec de procesul de asamblare (build) nu vor fi luate în considerare.

Mașina virtuală folosită pentru testarea temelor de casă pe vmchecker este descrisă în secțiunea Mașini virtuale din pagina de resurse.

Vă reamintim să parcurgeți atât secțiunea de depunctări cât și regulamentul de realizare a temelor.

Resurse

Scheletul și checker-ul sunt disponibile pe repository-ul de IOCLA.

Ultima modificare a checker-ului: 13.11.2020, ora 12:02

CC BY-SA CHIMERIC DE W3C CSS DOKUWIKI GET FIREFOX RSS XML FEED W3C XHTML 1.0