Wrocław, 15.03.2021r.

Architektura komputerów 2- laboratoria	Tomasz Bober nr indeksu: 252822
Grupa: **E07-00f**	Laboratoria 1: implementacja algorytmu rot13

1. Cel zajęć:

Implementacja algorytmu ROT13 zamieniającego małe litery na małe, a duże na duże. Maksymalna wielkość bufora do którego użytkownik może wprowadzić tekst do zakodowania to 100 liter. Cały program powstaje w architekturze 32-bitowej na platformie Linux. Algorytm szyfruje jedynie litery z alfabetu łacińskiego, pozostałe znaki pozostają niezmienione.

2. Algorytm ROT13:

Algorytm ROT13 działa w następujący sposób:

- litery z pierwszej połowy alfabetu(od A do M) podczas szyfrowania zostają przesunięte o 13 pozycji w tył alfabetu czyli literę A zamieniamy na N, a B na O.

- litery z drugiej połowy alfabetu(od N do Z) podczas szyfrowania zostają przesunięte o 13 pozycji w przód alfabetu czyli literę N zamieniamy na A, a O na B.

- dla małych liter algorytm działa analogicznie

Możemy zaobserwować, że algorytm ROT13 dla alfabetu łacińskiego jest swoją własną funkcją odwrotną.

3. Opis implementacji:

Do implementacji algorytmu użyłem skoków:

ja- skok wykona się jeśli porównana wcześniej wartość w rejestrze jest większa

jae- skok wykona się jeśli porównana wcześniej wartość w rejestrze jest większa bądź równa

jl- skok wykona się jeśli porównana wcześniej wartość w rejestrze jest mniejsza

Wszystkie operacje były wykonywane w pętli aby zakodować cały ciąg znaków wprowadzony z klawiatury.

Wrocław, 15.03.2021r.

Poniższy warunek pozwala przeskoczyć jedno wykonanie pętli jeśli podany znak ma kod ASCII większy od 7A.

cmpb $0x7A, input(%esi) -porównujemy literę z wpisanego tekstu(kod ASCII)

ja backsmall – jeśli litera ma kod ASCII większy niż 7A to przeskakujemy pozostałe instrukcję

cmpb $0x41, input(%esi) – jeśli kod ASCII litery jest większy/ równy od 41 skaczemy do 'big'

jae big

backbig:

cmpb $0x61, input(%esi) -jeśli kod ASCII litery jest większy/ równy od 61 skaczemy do 'small'

jae small

backsmall:

inc %esi -zwiększamy rejestr esi o 1 (licznik pętli)

cmp %edi, %esi -porównanie jeśli licznik jest mniejszy od edi(długość wprowadzonego tekstu) to kontynuujemy pętle

jl encrypt

Szyfrowanie dużych liter:

big: -do tej instrukcji trafiają litery o kodzie ASCII >=41

cmpb $0x4E, input(%esi) -jeśli kod ASCII litery jest<4E przeskocz do bigf w przeciwnym razie skok do bigs

jl bigf

jae bigs

jmp backbig

szyfrowanie pierwszej połowy alfabetu dla dużych liter

bigf:

addb $0x0D, input(%esi) -dodanie 13 czyli połowy długości alfabetu

jmp backbig -powrót do głównej pętli

szyfowanie drugiej połowy alfabetu dla dużych liter

bigs:

cmpb $0x5A, input(%esi) -jeśli kod ASCII litery jest większy od 5A wracamy do głównej pętli w przeciwnym razie odejmujemy 13 od kodu ASCII litery i wracamy do głównej pętli

ja backbig

subb $0x0D, input(%esi)

jmp backbig

Wczytywanie wejścia:

Aby wczytać tekst z klawiatury na początku tworzymy miejsce w którym będziemy go przechowywać

input: .space BUFFOR_LENGTH

gdzie BUFFOR_LENGTH to stała określająca maksymalną długość wprowadzanego tekstu:

BUFFOR_LENGTH= 100

Teraz wystarczy wywołać funkcję read:

Etykiety i ich wartości:

SYSREAD= 3, STDIN= 0

Funkcja:

mov $SYSREAD, %eax -wywołanie odpowiedniej funkcji w tym przypadku read
mov $STDIN, %ebx -określenie dokonywanej operacji
mov $input, %ecx -określenie w którym miejscu zapisać wczytane dane
mov $BUFFOR_LENGTH, %edx -określenie ilości miejsca przeznaczonego na wczytanie

Wyświetlenie na ekranie:

Wyświetlenie tekstu na ekranie jest bardzo podobne do wczytywania. Na początku określamy miejsce w którym znajduje się tekst oraz jego długość.

msg: .ascii "Write text you want to encrypt:\n" -podanie wiadomości do wyświetlenia
msg_len= . -msg -od końca tekstu odejmowany jest jego początek i tak uzyskujemy długość tekstu

Teraz możemy wywołać funkcję write:

Etykiety I ich wartości:

SYSWRITE= 4, STDOUT= 1

Funkcja:

mov $SYSWRITE, %eax -wywołanie odpowiedniej funkcji write
mov $STDOUT, %ebx -określenie dokonywanej operacji
mov $msg, %ecx -określenie miejsca w którym znajduje się tekst
mov $msg_len, %edx -określenie jak długi jest tekst który wypisujemy

Wrocław, 15.03.2021r.

4. Wnioski

Podczas laboratoriów nauczyłem się wczytywać oraz wypisywać tekst, porównywać wartości przy pomocy funkcji cmp, różnych rodzajów skoków tj. ja, jae, jl.
Program prawidłowo szyfruje wprowadzony tekst. Na początku jednak program nieprawidłowo szyfrował polskie znaki, jednakże ten błąd został usunięty poprzez wprowadzenie dodatkowego warunku sprawdzającego czy wprowadzony znak nie znajduje poza zakresem alfabetu łacińskiego w kodach ASCII.