Architektura komputerów

Laboratorium 1 w terminie 15 marca 2021

Jakub Superczyński 241381

28 marca 2021

1 Cel laboratorium

Celem laboratorium było napisanie w języku assemblera w architekturze 32-bitowej na platformę Linux programu wczytującego tekst ze standardowego wejścia (zakładając bufor na 100 znaków), realizującego na tym tekście szyfr ROT13 a następnie wypisującego dany zaszyfrowany już tekst.

2 Opis szyfru

Szyfr ROT13 to szyfr Cezara o 13-krotnym przesunięciu, to znaczy dana litera po zaszyfrowaniu staje się literą o 13 dalszą w kolejności alfabetycznej. Szyfr zmienia tylko znaki łacińskie, czyli zarówno znaki diakrytyczne jak i wszelkie znaki interpunkcyjne czy formatujące pozostają bez zmian. Dodatkowo istotne jest, że wielkie litery pozostają wielkimi literami, a małe - małymi.

3 Opis implementacji

Program można podzielić na dwa elementy:

3.1 Wczytywanie tekstu i wypisywanie tekstu

W programie raz wczytujemy tekst oraz dwukrotnie wypisujemy na ekran. Zadania te realizowane są analogicznie: do rejestru eax zapisujemy 4 dla wypisywania (wywołanie systemowe write) lub 3 (wywołanie systemowe read), do ebx zapisujemy 1 (jest to deskryptor standardowego wyjścia/wejścia), do ecx adres tekstu lub adres pamięci do której chcemy zapisać tekst, a do edx długość tekstu. Następnie wywołujemy przerwanie systemowe.

3.2 Algorytm szyfrowania

Wczytany tekst zapisany jest w kodzie ASCII jako jeden bajt na znak co bardzo ułatwia nam pracę. Cały algorytm to pętla iterująca przez każdy bajt zapisanego

tekstu. Najpierw algorytm sprawdza czym jest dany znak. Jeśli ma on wartość mniejszą niż 65, jest pewne że nie jest to znak łaciński. Program przeskakuje więc do końca pętli. Jeśli ma mniejszą niż 90, traktujemy dany znak jako wielką literę. Jeśli ma większą, należy sprawdzić czy jest to mała litera. W tym celu sprawdzamy czy wartość mieści się w zakresie między 98 a 122 (łącznie). Jeśli tak, traktujemy ten znak jako małą literę łacińską. Jeśli nie - program przeskakuje do końca pętli.
Program nieco inaczej, choć analogicznie, działa dla wielkich i małych liter. Opiera się to jednak na tej samej idei: traktowaniu danego zestawu liter niejako jako oddzielny alfabet, sprawdzeniu która z kolei jest dana liczba poprzez odjęcie wartości pierwszej liczby (A lub a), następnie przesunięciu o 13 poprzez zwykłe dodanie 13 (pamiętając o "zawijaniu" alfabetu po ostatniej literze, realizujemy to zwykle za pomocą funkcji modulo 26, ja zastosowałem warunkowe odejmowanie liczby 26) a następnie sprawdzeniu jaką wartością ten znak jest zakodowany w ASCII poprzez dodanie wcześniej odjętej wartości (czyli kodu A lub a). Czyli algorytm jest dokładnie taki sam w obu przypadkach, różni się jednak odejmowanymi wartościami. Aby nieco skrócić kod, jednocześnie odejmuję i dodaję, ponieważ wartości te są stałe.
Końcówka pętli (do której przechodzimy gdy znak nie jest literą łacińską) to zwiększenie o jeden bajt adresu sprawdzanego znaku (aby przejśc do następnej iteracji), dekrementacja licznika oraz sprawdzenie czy licznik wynosi 0. Jeśli wynosi zero, oznacza to że przeszliśmy już cały tekst więc wychodzimy z pętli. Jeśli jest większy niż 0, wracamy do początku pętli.

4 Problemy podczas implementacji

Laboratoria te były moim pierwszym poważniejszym spotkaniem z językiem asemblera, tak więc już samo wypisywanie i wczytywanie stanowiło pewien problem. Pierwszym krokiem było więc zapoznanie się z kodami wywołań systemowych. To pozwoliło napisać program wypisujący tekst oraz zapisujący dane od użytkownika. Opracowanie algorytmu ROT13 było stosunkowo łatwe, jako że szyfr Cezara to dość często występujący algorytm w różnych zadaniach. Pewnym problemem było dostosowanie operacji arytmetycznych do języka asemblera, choć to udało się zrealizować dosyć szybko. Opracowanie ich pozwoliło mi zapoznać się praktycznie z adresowaniem. Większym problemem stanowiło wykorzystanie instrukcji warunkowych do określenia typu danego znaku. Udało się to jednak również zrealizować, co pozwoliło na zapoznanie się ze sterowaniem warunkowym poprzez skoki do instrukcji.