Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет

информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина «Конструирование программ»

**ОТЧЕТ**

к лабораторной работе №8

на тему:

**«**Интеграция ассемблерных прерываний в проекты на C**»**

БГУИР 1-40 04 01

|  |
| --- |
| Выполнил студент группы 253502  АЛЬХОВИК Данила Игоревич |
|  |
| (дата, подпись студента) |
| Проверил старший преподаватель кафедры проектирования информационно-компьютерных систем  ЯЧИН Николай Сергеевич |
|  |
| (дата, подпись преподавателя) |

Минск 2023

**Цель работы:** Вариант 2.

На стороне Assembler: Реализация алгоритма поточного шифрования RC4, используя assembler-функции.

**Генератор случайных чисел: На базе системного таймера или других источников энтропии для создания случайных паролей.**

Пользователь может сохранять пароли и другую конфиденциальную информацию. Данные хранятся в зашифрованном виде. Реализуйте функцию генерации случайных паролей. Тайм-аут бездействия: Если менеджер паролей открыт и не используется в течение заданного времени, автоматически блокируйте его.

На стороне C++:

Главное меню и пользовательский интерфейс. Управление функциями ассемблера: добавление, удаление и редактирование записей; генерация пароля; шифрование и дешифрование данных. Логика тайм-аута бездействия.

**Ход работы:** на рисунке 1 представлены изначальные значения регистров, на рисунке 2 – ход выполнения задачи, на рисунке 3 - соответствующие значения после выполнения программы.

Листинг 1 – Исходный код программы задания  
asm\_manager.asm

default rel

SECTION .data align=1 noexec ; section number 2, data

SECTION .bss align=1 noexec ; section number 3, bss

global generate\_password: function

global swap: function

global rc4\_init: function

global rc4\_crypt: function

extern \_\_stack\_chk\_fail

extern rand

extern srand

extern time

SECTION .text align=1 exec ; section number 1, code

generate\_password:; Function begin

push rbp

mov rbp, rsp

add rsp, -128

mov qword [rbp-78H], rdi

mov dword [rbp-7CH], esi

mov rax, qword [fs:abs 28H]

mov qword [rbp-8H], rax

xor eax, eax

mov rax, qword 6867666564636261H

mov rdx, qword 706F6E6D6C6B6A69H

mov qword [rbp-96H], rax

mov qword [rbp-58H], rdx

mov rax, qword 7877767574737271H

mov rdx, qword 4645444342417A79H

mov qword [rbp-50H], rax

mov qword [rbp-48H], rdx

mov rax, qword 4E4D4C4B4A494847H

mov rdx, qword 565554535251504FH

mov qword [rbp-40H], rax

mov qword [rbp-38H], rdx

mov rax, qword 333231305A595857H

mov rdx, qword 4021393837363534H

mov qword [rbp-30H], rax

mov qword [rbp-28H], rdx

mov rax, qword 29282A265E252423H

mov qword [rbp-20H], rax

mov word [rbp-18H], 11103

mov byte [rbp-16H], 0

mov dword [rbp-64H], 74

mov edi, 0

call time

mov edi, eax

call srand

mov dword [rbp-68H], 0

jmp startloop1

random:

call rand

cdq

idiv dword [rbp-64H]

mov ecx, edx

mov eax, dword [rbp-68H]

movsxd rdx, eax

mov rax, qword [rbp-78H]

add rdx, rax

movsxd rax, ecx

movzx eax, byte [rbp+rax-60H]

mov byte [rdx], al

add dword [rbp-68H], 1

startloop1:

mov eax, dword [rbp-68H]

cmp eax, dword [rbp-7CH]

jl random

mov eax, dword [rbp-7CH]

movsxd rdx, eax

mov rax, qword [rbp-78H]

add rax, rdx

mov byte [rax], 0

nop

mov rax, qword [rbp-8H]

sub rax, qword [fs:abs 28H]

jz gen\_end

call \_\_stack\_chk\_fail

gen\_end: leave

ret

; generate\_password End of function

swap: ; Function begin

push rbp

mov rbp, rsp

mov qword [rbp-18H], rdi

mov qword [rbp-20H], rsi

mov rax, qword [rbp-18H]

movzx eax, byte [rax]

mov byte [rbp-1H], al

mov rax, qword [rbp-20H]

movzx edx, byte [rax]

mov rax, qword [rbp-18H]

mov byte [rax], dl

mov rax, qword [rbp-20H]

movzx edx, byte [rbp-1H]

mov byte [rax], dl

nop

pop rbp

ret

; swap End of function

rc4\_init:; Function begin

push rbp

mov rbp, rsp

sub rsp, 40

mov qword [rbp-18H], rdi

mov dword [rbp-1CH], esi

mov qword [rbp-28H], rdx

mov dword [rbp-4H], 0

mov dword [rbp-8H], 0

jmp startloop2

loop2: mov eax, dword [rbp-8H]

movsxd rdx, eax

mov rax, qword [rbp-28H]

add rax, rdx

mov edx, dword [rbp-8H]

mov byte [rax], dl

add dword [rbp-8H], 1

startloop2: cmp dword [rbp-8H], 255

jle loop2

mov dword [rbp-8H], 0

jmp compareloop

cypher: mov eax, dword [rbp-8H]

movsxd rdx, eax

mov rax, qword [rbp-28H]

add rax, rdx

movzx eax, byte [rax]

movzx edx, al

mov eax, dword [rbp-4H]

lea ecx, [rdx+rax]

mov eax, dword [rbp-8H]

cdq

idiv dword [rbp-1CH]

mov eax, edx

movsxd rdx, eax

mov rax, qword [rbp-18H]

add rax, rdx

movzx eax, byte [rax]

movzx eax, al

add eax, ecx

cdq

shr edx, 24

add eax, edx

movzx eax, al

sub eax, edx

mov dword [rbp-4H], eax

mov eax, dword [rbp-4H]

movsxd rdx, eax

mov rax, qword [rbp-28H]

add rdx, rax

mov eax, dword [rbp-8H]

movsxd rcx, eax

mov rax, qword [rbp-28H]

add rax, rcx

mov rsi, rdx

mov rdi, rax

call swap

add dword [rbp-8H], 1

compareloop:

cmp dword [rbp-8H], 255

jle cypher

nop

nop

leave

ret

; rc4\_init End of function

rc4\_crypt:; Function begin

push rbp

mov rbp, rsp

sub rsp, 40

mov qword [rbp-18H], rdi

mov dword [rbp-1CH], esi

mov qword [rbp-28H], rdx

mov dword [rbp-10H], 0

mov dword [rbp-0CH], 0

mov dword [rbp-8H], 0

jmp comparloop3

startloop3: mov eax, dword [rbp-10H]

add eax, 1

cdq

shr edx, 24

add eax, edx

movzx eax, al

sub eax, edx

mov dword [rbp-10H], eax

mov eax, dword [rbp-10H]

movsxd rdx, eax

mov rax, qword [rbp-28H]

add rax, rdx

movzx eax, byte [rax]

movzx edx, al

mov eax, dword [rbp-0CH]

add eax, edx

cdq

shr edx, 24

add eax, edx

movzx eax, al

sub eax, edx

mov dword [rbp-0CH], eax

mov eax, dword [rbp-0CH]

movsxd rdx, eax

mov rax, qword [rbp-28H]

add rdx, rax

mov eax, dword [rbp-10H]

movsxd rcx, eax

mov rax, qword [rbp-28H]

add rax, rcx

mov rsi, rdx

mov rdi, rax

call swap

mov eax, dword [rbp-10H]

movsxd rdx, eax

mov rax, qword [rbp-28H]

add rax, rdx

movzx edx, byte [rax]

mov eax, dword [rbp-0CH]

movsxd rcx, eax

mov rax, qword [rbp-28H]

add rax, rcx

movzx eax, byte [rax]

add eax, edx

movzx eax, al

mov dword [rbp-4H], eax

mov eax, dword [rbp-8H]

movsxd rdx, eax

mov rax, qword [rbp-18H]

add rax, rdx

movzx ecx, byte [rax]

mov eax, dword [rbp-4H]

movsxd rdx, eax

mov rax, qword [rbp-28H]

add rax, rdx

movzx edx, byte [rax]

mov eax, dword [rbp-8H]

movsxd rsi, eax

mov rax, qword [rbp-18H]

add rax, rsi

xor edx, ecx

mov byte [rax], dl

add dword [rbp-8H], 1

comparloop3:

mov eax, dword [rbp-8H]

cmp eax, dword [rbp-1CH]

jl startloop3

nop

nop

leave

ret

; rc4\_crypt End of function

manager.c

extern void generate\_password(char \*password, int length);

extern void swap(unsigned char \*a, unsigned char \*b);

extern void rc4\_init(unsigned char \*key, int key\_length, unsigned char \*S);

extern void rc4\_crypt(unsigned char \*data, int data\_length, unsigned char \*S);  
  
 // Load data from file into array of structures

FILE \*file = fopen("data.txt", "r");

if (file != NULL)

{

while (num\_records < MAX\_RECORDS && fscanf(file, "%s %s %s", records[num\_records].name, records[num\_records].username, records[num\_records].password) == 3) {

unsigned char S[256];

char key[] = "mysecretkey";

rc4\_init(key, strlen(key), S);

rc4\_crypt(records[num\_records].name, strlen(records[num\_records].name), S);

rc4\_init(key, strlen(key), S);

rc4\_crypt(records[num\_records].username, strlen(records[num\_records].username), S);

rc4\_init(key, strlen(key), S);

rc4\_crypt(records[num\_records].password, strlen(records[num\_records].password), S);

num\_records++;

}

fclose(file);

}

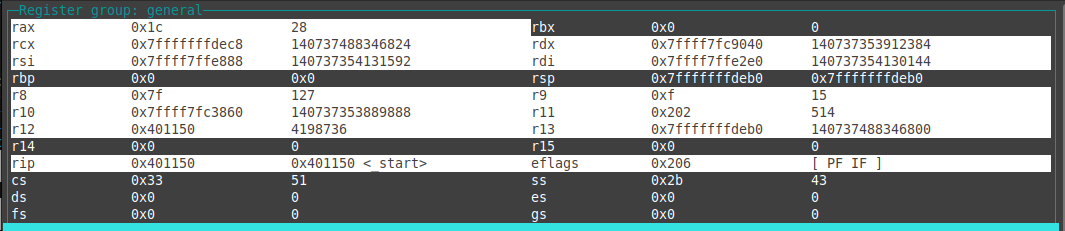


Рисунок 1 – Значения регистров перед выполнением

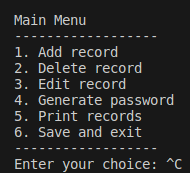


Рисунок 2 – Ход выполнения задачи

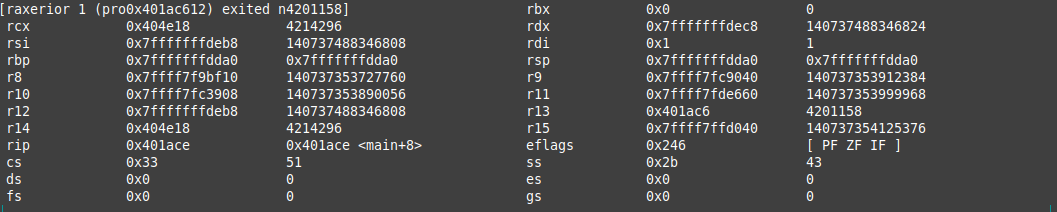


Рисунок 3 – Значения регистров после выполнения