

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение
высшего образования «Санкт-Петербургский политехнический университет
Петра Великого»

Институт компьютерных наук и кибербезопасности

Направление: 02.03.01 Математика и компьютерные науки

Отчет по дисциплине: «Основы архитектуры ЦВМ»

Система команд, способы адресации и использование подпрограмм.

Студент,
группы 5130201/40003

_____ Адиатуллин Т. Р

Руководитель,
Преподаватель

_____ Вербова Н. М.

«____» _____ 2025 г.

Санкт-Петербург, 2025

Содержание

1 Цель работы	3
2 Методика	3
3 Исходный код на языке Си	3
4 Дизассемблированный код	3
5 Соглашения о вызовах (Calling Conventions)	4
6 Вывод	6

1 Цель работы

Изучить ассемблерный код программы с вызовом функции.

2 Методика

1. Написать программу на языке Си с учетом спецификации функции, в зависимости от варианта.
2. Оттранслировать программу, содержащую этот фрагмент.
3. Перейти в режим отладки и изучить дизассемблированный код фрагмента, про- комментировать все команды.

3 Исходный код на языке Си

Вариант 3-10:

```
1 long lA;
2 char cB;
3 char strC[20];
4
5 long Fn(char, char*);
6
7 int main(void) {
8     cB = 'A';
9     lA = Fn(cB, strC);
10    return 0;
11 }
12
13 long Fn(char chParam, char* strParam) {
14     int iLoc;
15     char cLoc;
16
17     iLoc = (int)chParam + strParam[0];
18     cLoc = (char)(iLoc % 8);
19
20     return (long)(iLoc + cLoc);
21 }
```

Листинг 1: Исходный код программы на языке С

4 Дизассемблированный код

Дизассемблированный код представлен ниже:

```

00000000 <_main>:
; int main(void) {
    push    ebp          ; сохранение базового указателя
    mov     ebp, esp      ; установка нового базового указателя
    sub     esp, 0x18       ; выделение места под локальные переменные
    mov     dword ptr [ebp-0x4], 0x0 ; инициализация локальной переменной
;   cB = 'A';
    mov     byte ptr [0x0], 0x41 ; записываю символ 'A' в память
;   lA = Fn(cB, strC);
    lea     eax, [0x0]      ; загружаю адрес строки
    movsx  ecx, byte ptr [0x0] ; беру символ и расширяю до int
    mov     dword ptr [esp], ecx ; кладу первый аргумент на стек
    mov     dword ptr [esp+0x4], eax ; кладу второй аргумент (указатель) на стек
    call   0x40 <_Fn>        ; вызов функции Fn
    mov     dword ptr [0x0], eax ; сохраняю результат
;   return 0;
    xor    eax, eax        ; возвращаемое значение 0
    add    esp, 0x18        ; очистка стека
    pop    ebp            ; восстановление базового указателя
    ret                    ; возврат из main

00000040 <_Fn>:
; long Fn(char chParam, char* strParam) {
    push    ebp          ; сохранение базового указателя
    mov     ebp, esp      ; установка нового базового указателя
    sub     esp, 0x8       ; выделение места под локальные переменные
;   iLoc = (int)chParam + strParam[0];
    mov     eax, dword ptr [ebp+0xc] ; загружаю указатель
    mov     al, byte ptr [ebp+0x8] ; загружаю символ
    movsx  ecx, byte ptr [ebp+0x8] ; расширяю символ до int
    mov     edx, dword ptr [ebp+0xc] ; беру указатель на строку
    movsx  ecx, byte ptr [ecx]      ; беру первый символ строки
    add    eax, ecx            ; складываю символ и первый элемент строки
    mov     dword ptr [ebp-0x4], eax ; сохраняю в iLoc
;   cLoc = (char)(iLoc % 8);
    mov     eax, dword ptr [ebp-0x4] ; загружаю iLoc
    mov     ecx, 0x8           ; делитель 8
    cdq                  ; расширяю eax в edx:eax
    idiv   ecx              ; делию iLoc на 8
    mov     al, dl             ; беру остаток из dl
    mov     byte ptr [ebp-0x5], al ; сохраняю в cLoc
;   return (long)(iLoc + cLoc);
    mov     eax, dword ptr [ebp-0x4] ; загружаю iLoc
    movsx  ecx, byte ptr [ebp-0x5] ; загружаю cLoc
    add    eax, ecx            ; складываю iLoc и cLoc
    add    esp, 0x8            ; очистка стека
    pop    ebp            ; восстановление базового указателя
    ret                    ; возврат из функции Fn

```

Рис. 1: Дизассемблированный код

5 Соглашения о вызовах (Calling Conventions)

Определение

Соглашение о вызовах — это набор правил, определяющих:

- как передаются аргументы функции (через стек или регистры);
 - кто отвечает за очистку стека (вызывающая или вызываемая функция);
 - в каком регистре возвращается результат;
 - какие регистры должны сохраняться при вызове.

Основные соглашения в x86 (32-bit)

1. cdecl (C declaration)

- Аргументы передаются через стек (справа налево).
- Очистку стека выполняет вызывающая функция.
- Возвращаемое значение — в регистре **EAX**.
- Используется по умолчанию в C/C++ на Unix/Linux, macOS.

2. stdcall (standard call)

- Аргументы передаются через стек.
- Очистку стека выполняет вызываемая функция.
- Возвращаемое значение — в **EAX**.
- Используется в WinAPI.

3. fastcall

- Первые аргументы передаются через регистры (**ECX**, **EDX**), остальные — через стек.
- Очистку стека выполняет вызываемая функция.
- Возвращаемое значение — в **EAX**.

4. thiscall

- Используется для методов классов в C++.
- Указатель **this** передаётся через **ECX**.
- Остальные аргументы — через стек.
- Возвращаемое значение — в **EAX**.

Соглашение в лабораторной работе

В приведённом дизассемблированном коде видно:

- Аргументы кладутся на стек:

```
mov dword ptr [esp], ecx      ; первый аргумент  
mov dword ptr [esp+0x4], eax ; второй аргумент  
call 0x40 <_Fn>           ; вызов функции Fn
```

- После возврата стек очищается в вызывающей функции:

```
add esp, 0x18                ; очистка стека
```

- Возвращаемое значение приходит в ЕАХ:

```
mov dword ptr [0x0], eax     ; сохранение результата
```

Это соответствует соглашению **cdecl**:

- аргументы передаются через стек;
- очистку стека делает вызывающая функция;
- результат возвращается через ЕАХ.

В лабораторной работе используется соглашение о вызовах **cdecl**, характерное для компиляторов С на системах Unix/macOS и применяемое по умолчанию.

6 Вывод

Был дизассемблирован исходный код. Были изучены и разобраны команды ди-зассемблированного кода, разобраны особенности кода с вызовом функций.