РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ

ЛАБОРАТОРНАЯ РАБОТА № 7

дисциплина: Архитектура компьютера

Студент: Мизинов М.Г.

Группа: НКАбд-04-25

№ ст. билета: 1032253540

МОСКВА

2025 г.

СОДЕРЖАНИЕ

Список иллюстраций

Список таблиц

Основная часть

1. Цель работы

Изучение команд условного и безусловного переходов. Приобретение навыков написания программ с использованием переходов. Знакомство с назначением и структурой файла листинга.

2. Теоретическое введение

Для реализации ветвлений в ассемблере используются так называемые команды передачи управления или команды перехода. Можно выделить 2 типа переходов:

- условный переход – выполнение или не выполнение перехода в определенную точку программы в зависимости от проверки условия.
- безусловный переход – выполнение передачи управления в определенную точку программы без каких-либо условий.

3. Задание

На основе методических указаний провести ознакомительную работу с командами безусловного и условного переходов в Nasm.

4. Выполнение лабораторной работы

4.1 Реализация переходов в NASM

Создал папку lab7 и файл asm (рис. 1).

```
mgmizinov@mint:~$ cd ~/Mizinov-study_2025-2026_arh-pc/labs/
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs$ mkdir lab7
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs$ cd lab7
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab7$ touch lab7-1.asm
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab7$
```

Рис. 1: Создаю начальные условия

Код файла lab7-1.asm (рис. 2).

```
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0
SECTION .text
GLOBAL _start
_start:
jmp _label2
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 1'
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 2'
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 3'
_end:
call quit ; вызов подпрограммы завершения
```

Рис. 2: Файл lab7-1.asm

Запуск программы lab7-1.asm (рис. 3).

```
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab7$ nasm -f elf lab7-1.asm
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab7$ ld -m elf_i386 -o lab7-1 lab7-1.o
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab7$ ./lab7-1
Сообщение № 2
Сообщение № 3
```

Рис. 3: Выполнение lab6-1.asm

Изменим код программы (рис. 3).

```
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0
SECTION .text
GLOBAL _start
_start:
jmp _label2
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 1'
jmp _end
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 2'
jmp _label1
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 3'
_end:
call quit ; вызов подпрограммы завершения
```

Рис. 4: Изменённый lab7-1.asm

А затем снова запустим файл (рис. 5).

```
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab7$ nasm -f elf lab7-1.asm
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab7$ ld -m elf_i386 -o lab7-1 lab7-1.o
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab7$ ./lab7-1
Сообщение № 2
Сообщение № 1
```

Рис. 5: Запуск изменённого lab7-1.asm

Теперь снова изменим файл чтобы строки выводились в нужном порядке (рис. 6).

```
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0

SECTION .text
GLOBAL _start
_start:

jmp _label3

_label1:
mov eax, msg1
call sprintLF
jmp _end

_label2:
mov eax, msg2
call sprintLF
jmp _label1

_label3:
mov eax, msg3
call sprintLF
jmp _label2

_end:
call quit
```

Рис. 6: Дважды изменённый lab7-1.asm

Запуск изменённой дважды программы lab7-1.asm (рис. 7).

```
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab7$ nasm -f elf lab7-1.asm
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab7$ ld -m elf_i386 -o lab7-1 lab7-1.o
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab7$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab7$
```

Рис. 7: Запуск дважды изменённого lab7-1.asm

Создадим программу lab7-2.asm (рис. 8).

```
%include 'in_out.asm'
section .data
msg1 db 'Введите B: ',0h
msg2 db "Наибольшее число: ",0h
A dd '20'
C dd '50'
section .bss
max resb 10
B resb 10
section .text
global _start
_start:
; ---------- Вывод сообщения 'Введите B: '
mov eax,msg1
call sprint
; ---------- Ввод 'B'
mov ecx,B
mov edx,10
call sread
; ---------- Преобразование 'B' из символа в число
mov eax,B
call atoi ; Вызов подпрограммы перевода символа в число
mov [B],eax ; запись преобразованного числа в 'B'
; ---------- Записываем 'A' в переменную 'max'
mov ecx,[A] ; 'ecx = A'
mov [max],ecx ; 'max = A'
; ---------- Сравниваем 'A' и 'C' (как символы)
cmp ecx,[C] ; Сравниваем 'A' и 'C'
jg check_B ; если 'A>C', то переход на метку 'check_B',
mov ecx,[C] ; иначе 'ecx = C'
mov [max],ecx ; 'max = C'
; ---------- Преобразование 'max(A,C)' из символа в число
check_B:
mov eax,max
call atoi ; Вызов подпрограммы перевода символа в число
mov [max],eax ; запись преобразованного числа в `max`
; ---------- Сравниваем 'max(A,C)' и 'B' (как числа)
mov ecx,[max]
cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'
jg fin ; если 'max(A,C)>B', то переход на 'fin',
mov ecx,[B] ; иначе 'ecx = B'
mov [max],ecx
; ---------- Вывод результата
fin:
mov eax, msg2
call sprint ; Вывод сообщения 'Наибольшее число: '
mov eax,[max]
call iprintLF ; Вывод 'max(A,B,C)'
call quit ; Выход
```

Рис. 8: Создание lab7-2.asm

А затем запустим её (рис. 9).

```
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab7$ touch lab7-2.asm
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab7$ nasm -f elf lab7-2.asm
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab7$ ld -m elf_i386 -o lab7-2 lab7-2.o
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab7$ ./lab7-2
Введите B: 3
Наибольшее число: 50
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab7$ ./lab7-2
Введите B: 25
Наибольшее число: 50
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab7$
```

Рис. 9: Запуск lab7-2.asm

4.2 Изучение структуры файлы листинга

Ознакомимся с файлом lab7-2.lst, его форматом и содержимым (рис. 10).

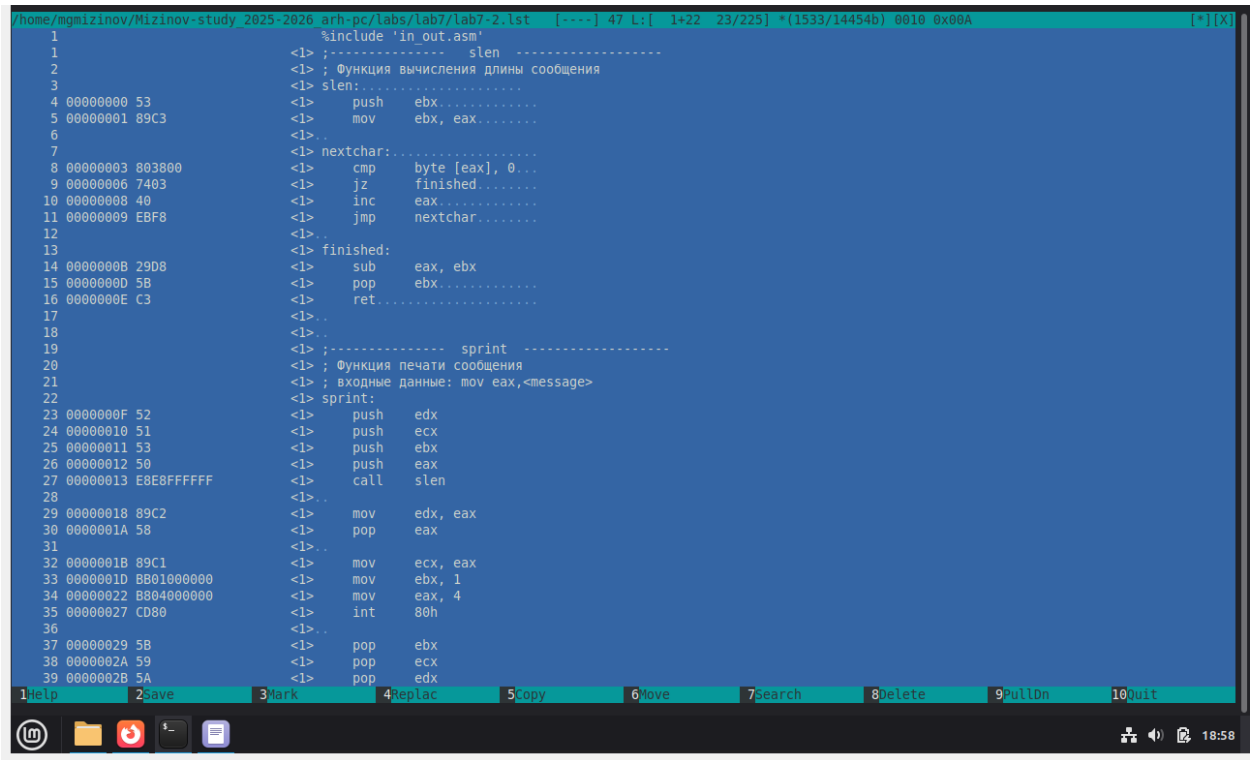

Рис. 10: Изучение lab7-2.lst

Формат строки в файле листинга такой: первым делом идёт её внутренний номер (не обязательно тот же, что в исходном файле). Далее указывается адрес — то есть, смещение машинного кода в сегменте. После этого следует сам машинный код, а завершает всё исходный текст программы, включая комментарии.

Удаляю один операнд, чтобы проверить поведение файла листинга в дальнейшем (рис. 11).

```
; ---------- Преобразование 'max
check_B:
mov eax,
call atoi ; Вызов подпрограммы п
mov [max],eax ; запись преобразо
; ---------- Сравниваем 'max(A,C
```

Рис. 11: Изменённый lab7-2.asm

Проверка изменённой программы lab7-2.asm (рис. 12).

```
 33 0000012A 890D[00000000]          mov [max],ecx ; 'max = C'
 34                                  ; ---------- Преобразование 'max(A,C)' из символа в число
 35                                  check_B:
 36                                  mov eax,
 36          ******************       error: invalid combination of opcode and operands
 37 00000130 E867FFFFFF              call atoi ; Вызов подпрограммы перевода символа в число
 38 00000135 A3[00000000]            mov [max],eax ; запись преобразованного числа в `max`
 39                                  ; ---------- Сравниваем 'max(A,C)' и 'B' (как числа)
 40 0000013A 8B0D[00000000]          mov ecx,[max]
```

Рис. 12: Проверка изменённого lab7-2.asm

Как и ожидалось программа пишет ошибку.

5. Задание для самостоятельной работы

У меня вариант номер 1, как я выяснил во время 6й лабораторной работы. А значит использую следующие данные:

Значения a, b, c:

17,23,45

Выражение для $f(x)$

{ 2a-x, x<a

{

{ 8, x>=a

X1: 1

A1: 2

X2: 2

A2: 1

Реализуем файл code1.asm для первого индивидуального задания (рис. 13).

```
%include "in_out.asm"

section .data
    a dd 17
    b dd 23
    c dd 45

section .text
global _start

_start:
    mov eax, [a]    ; Берем a
    cmp eax, [b]    ; Сравниваем с b
    jle check_c     ; Если a <= b
    mov eax, [b]    ; Иначе берем b

check_c:
    cmp eax, [c]    ; Сравниваем с c
    jle print       ; Если текущее <= c
    mov eax, [c]    ; Иначе берем c

print:
    call iprintLF   ; Выводим результат с переводом строки
    call quit       ; Завершаем программу
```

Рис. 13: Создание code1.asm

Запуск программы code1.asm и проверка табличных значений (рис. 14).

```
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab7$ nasm -f elf code1.asm
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab7$ ld -m elf_i386 -o code1 code1.o
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab7$ ./code1
17
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab7$
```

Рис. 14: Выполнение code1.asm

Реализуем файл code2.asm для второго индивидуального задания (рис. 15).

```
%include "in_out.asm"

SECTION .data
msg_x: DB 'Введите x: ',0
msg_a: DB 'Введите a: ',0
msg_result: DB 'Результат: ',0

SECTION .bss
x: RESB 80
a: RESB 80

SECTION .text
GLOBAL _start

; Функция f(x)
f_x:
    cmp ebx, ecx        ; Сравниваем x и a
    jge .else_case      ; если x >= a

    ; Случай x < a: f(x) = 2a - x
    mov eax, ecx        ; eax = a
    add eax, eax        ; eax = 2a
    sub eax, ebx        ; eax = 2a - x
    ret

.else_case:
    ; Случай x >= a: f(x) = 8
    mov eax, 8
    ret

_start:
    ; Ввод x
    mov eax, msg_x
    call sprint

    mov ecx, x
    mov edx, 80
    call sread
    mov eax, x
    call atoi           ; eax = x
    push eax            ; сохраняем x в стек

    ; Ввод a
    mov eax, msg_a
    call sprint

    mov ecx, a
    mov edx, 80
    call sread
    mov eax, a
    call atoi           ; eax = a
    mov ecx, eax        ; ecx = a
    pop ebx             ; ebx = x

    ; Вызов функции f(x)
    call f_x

    ; Вывод результата
    push eax
    mov eax, msg_result
    call sprint
    pop eax
    call iprintLF

    call quit
```

Рис. 15: Создание code2.asm

Запуск программы code2.asm (рис. 16).

```
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab7$ nasm -f elf code2.asm
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab7$ ld -m elf_i386 -o code2 code2.o
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab7$ ./code2
Введите x: 1
Введите a: 2
Результат: 3
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab7$ ./code2
Введите x: 2
Введите a: 1
Результат: 8
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab7$
```

Рис. 16: Выполнение code2.asm

Ссылка на github: https://github.com/MihailMizinov/Mizinov-study_2025-2026_arh-pc

Выводы

При выполнении данной лабораторной работы я приобрёл практические навыки работы с командами безусловного и условного переходов в Nasm.

Список литературы

1) Лабораторная работа №7. https://esystem.rudn.ru/pluginfile.php/2089087/mod_resource/content/0/%D0%9B%D0%B0%D0%B1%D0%BE%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%BD%D0%B0%D1%8F%20%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%20%E2%84%967.%20%D0%9A%D0%BE%D0%BC%D0%B0%D0%BD%D0%B4%D1%8B%20%D0%B1%D0%B5%D0%B7%D1%83%D1%81%D0%BB%D0%BE%D0%B2%D0%BD%D0%BE%D0%B3%D0%BE%20%D0%B8%20%D1%83%D1%81%D0%BB%D0%BE%D0%B2%D0%BD%D0%BE%D0%B3%D0%BE%20%D0%BF%D0%B5%D1%80%D0%B5%D1%85%D0%BE%D0%B4%D0%BE%D0%B2%20%D0%B2%20Nasm.%20%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%20%D0%B2%D0%B5%D1%82%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B9..pdf

2) Википедия. https://en.wikipedia.org/wiki/GitHub