Отчет по лабораторной работе №7

Дисциплина: Архитектура компьютера

Челухаев Кирилл Александрович

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

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

2 Задания

1. Реализация переходов в NASM
2. Изучение структуры файлы листинга
3. Задание для самостоятельной работы

3 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.

4 Выполнение лабораторной работы

Я создал каталог lab07 для лабораторной работы №7 и создал в нем файл lab7-1.asm.(рис. **??**).

```
kachelukhaev1@dk2n23 ~/work/study/2024-2025/Архитектура компьютера/arch-pc $
mkdir lab07
kachelukhaev1@dk2n23 ~/work/study/2024-2025/Архитектура компьютера/arch-pc $
cd lab07
kachelukhaev1@dk2n23 ~/work/study/2024-2025/Архитектура компьютера/arch-pc/la
b07 $ touch lab7-1.asm
kachelukhaev1@dk2n23 ~/work/study/2024-2025/Архитектура компьютера/arch-pc/la
b07 $
```

Я ввел в файл код программы из ТУИС, скомпилировал исполняемый файл и запустил его.(рис. **??**).

```
b07 $ nasm -f elf lab7-1.asm
kachelukhaev1@dk2n23 ~/work/study/2024-2025/Архитектура компьютера/arch-pc/la
b07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
kachelukhaev1@dk2n23 ~/work/study/2024-2025/Архитектура компьютера/arch-pc/la
b07 $ ./lab7-1
Сообщение № 2
Сообщение № 3
kachelukhaev1@dk2n23 ~/work/study/2024-2025/Архитектура компьютера/arch-pc/la
b07 $
```

Далее я изменил текст программы добавив или изменив инструкции jmp, чтобы вывод программы был следующим:

Сообщение № 3 Сообщение № 2 *Сообщение № 1

Я скомпилировал исполняемый файл и проверил его работу (рис. **??**).

```
kachelukhaev1@dk2n23 ~/work/study/2024-2025/Архитектура компьютера/arch-pc/la
b07 $ nasm -f elf lab7-1.asm
kachelukhaev1@dk2n23 ~/work/study/2024-2025/Архитектура компьютера/arch-pc/la
b07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
kachelukhaev1@dk2n23 ~/work/study/2024-2025/Архитектура компьютера/arch-pc/la
b07 $ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
kachelukhaev1@dk2n23 ~/work/study/2024-2025/Архитектура компьютера/arch-pc/la
b07 $
```

Далее я создал файл lab7-2.asm и ввел в него код программы, которая опреде-

ляет и выводит на экран наибольшую из 3 целочисленных переменных: A,B и C. Создал исполняемый файл и проверил его работу для разных значений B. (рис. **??**).

```
b07 $ touch lab7-2.asm
kachelukhaev1@dk2n23 ~/work/study/2024-2025/Архитектура компьютера/arch-pc/la
b07 $ nasm -f elf lab7-2.asm
kachelukhaev1@dk2n23 ~/work/study/2024-2025/Архитектура компьютера/arch-pc/la
b07 $ ld -m elf_i386 -o lab7-2 lab7-2.o
kachelukhaev1@dk2n23 ~/work/study/2024-2025/Архитектура компьютера/arch-pc/la
b07 $ ./lab7-2
Введите B: 3
Наибольшее число: 50
kachelukhaev1@dk2n23 ~/work/study/2024-2025/Архитектура компьютера/arch-pc/la
b07 $ ./lab7-2
Введите B: 51
Наибольшее число: 51
kachelukhaev1@dk2n23 ~/work/study/2024-2025/Архитектура компьютера/arch-pc/la
b07 $
```

4.1 Изучение структуры файлы листинга

Обычно nasm создаёт в результате ассемблирования только объектный файл. Получить файл листинга можно, указав ключ -l и задав имя файла листинга в командной строке. Я создал файл листинга для программы из файла lab7-2.asm и открыл его. (рис. 4.1).

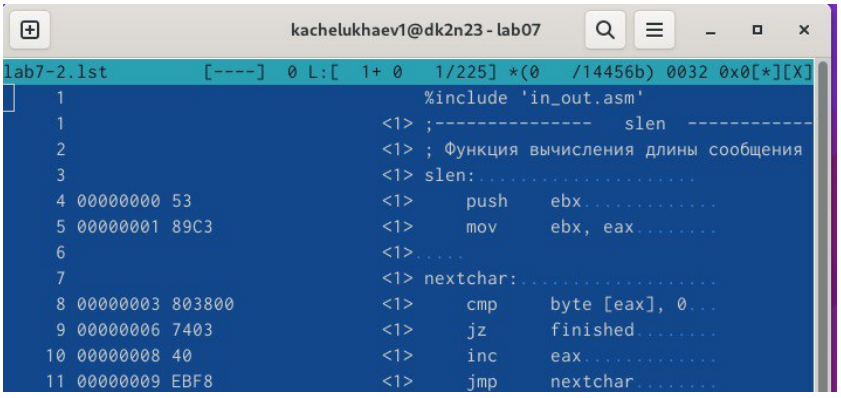

Рис. 4.1: 5

1. Строка 4: 4 00000000 53 <1> push ebx

- Адрес: 00000000 — это первый байт подпрограммы. Код будет помещён в память, начиная с этого адреса

- Машинный код: 53 — это шестнадцатеричное представление машинного кода для инструкции push ebx. Инструкция push помещает содержимое регистра ebx в стек.
- Комментарии: <1> указывают на тип элемента в листинге. В данном случае это просто инструкция на ассемблере
- Действие: сохраняет текущее значение регистра ebx в стеке. Это необходимо для сохранения значения регистра ebx, которое будет использоваться в этой подпрограмме.

2. Строка 5: 5 00000001 89C3 <1> mov ebx, eax

- Адрес: 00000001 — инструкция будет помещена по этому адресу, который на 1 байт больше, чем у предыдущей инструкции.
- Машинный код: 89C3 — это машинный код для инструкции mov ebx, eax
- Комментарии: <1> — Указывает, что это ассемблерная инструкция.
- Действие: перемещает содержимое регистра eax в регистр ebx.

3. Строка 8: 8 00000003 803800 <1> cmp byte [eax], 0

- Адрес: 00000003 - Адрес памяти для этой инструкции.
- Машинный код: 803800 - Машинный код для инструкции cmp byte [eax], 0.
- Комментарии: <1> - Указывает, что это ассемблерная инструкция.
- Действие: Сравнивает байт по адресу, указанному в eax, с нулем. Это нужно для определения конца строки.

4.2 Задание для самостоятельной работы

1. Напишите программу нахождения наименьшей из 3 целочисленных переменных a b c Значения переменных выбрать из табл. 7.5 в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу.

Я создал файл lab7-3.asm и написал в нем код для программы нахождения наименьшей из 3 целочисленных переменных a b c и проверил его работу (рис. 4.2).

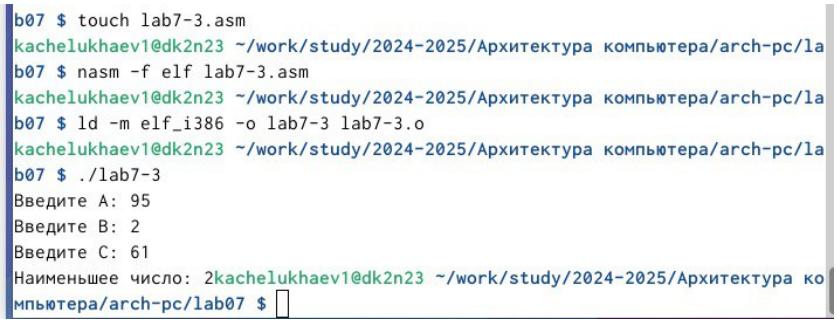

Рис. 4.2: 6

```
%include 'in_out.asm'  ; Подключение внешнего файла

SECTION .data
    msg1:   db 'Введите A: ', 0
    msg2:   db 'Введите B: ', 0
    msg3:   db 'Введите C: ', 0
    msg_res: db 'Наименьшее число: ', 0
    A:      dd 0  ; Переменные для хранения A, B, и C
    B:      dd 0
    C:      dd 0

SECTION .bss
```

```
    min:    resd 4  ; Переменная для хранения наименьшего числа

SECTION .text
    global _start

_start:
    ; ---- Ввод A ----
    mov eax, msg1    ; Вывод сообщения "Введите A:"
    call sprint
    mov ecx, A       ; Адрес переменной A
    mov edx, 10      ; Максимальная длина ввода
    call sread        ; Ввод значения
    mov eax, A
    call atoi        ; Преобразование строки в целое, результат в EAX
    mov [A], eax       ; Записываем преобразованное число в A

    ; ---- Ввод B ----
    mov eax, msg2    ; Вывод сообщения "Введите B:"
    call sprint
    mov ecx, B       ; Адрес переменной B
    mov edx, 10      ; Максимальная длина ввода
    call sread        ; Ввод значения
    mov eax, B
    call atoi        ; Преобразование строки в целое, результат в EAX
    mov [B], eax       ; Записываем преобразованное число в B

     ; ---- Ввод C ----
    mov eax, msg3    ; Вывод сообщения "Введите C:"
    call sprint
```

```
mov ecx, C        ; Адрес переменной C
mov edx, 10       ; Максимальная длина ввода
call sread         ; Ввод значения
mov eax, C
call atoi         ; Преобразование строки в целое, результат в EAX
mov [C], eax        ; Записываем преобразованное число в C

; ---- Сравнение A и B ----
mov eax, [A]      ; Загрузка значения A в eax
mov ebx, [B]      ; Загрузка значения B в ebx
cmp eax, ebx       ; Сравнение A и B
jle check_C       ; Если A <= B, перейти к сравнению с C
mov eax, ebx       ; Если A > B, то  eax = B
; иначе  eax = A
mov [min],eax       ; Сохраняем меньшее из A и B в min
jmp check_C_2     ; Пропустить следующую строчку, если A<=B

check_C:
  mov eax,[A]     ; eax = A
mov [min],eax     ; Сохраняем меньшее из A и B в min


 check_C_2:
; ---- Сравнение min и C ----
mov eax, [min]    ; Загрузка значения min(A,  в eax
mov ebx, [C]      ; Загрузка значения C в ebx
cmp eax, ebx       ; Сравнение min(A,и C
jle fin           ; Если min(A, <= C, перейти к fin
mov eax, ebx       ; Если min(A,B) > C, то min(A,B,C) = C
```

```
mov [min],eax
; иначе min(A,B,C)=min(A,B)

; ---- Вывод результата ----
fin:
    mov eax, msg_res ; Вывод сообщения "Наименьшее число: "
    call sprint
    mov eax, [min] ; Загрузка наименьшего числа из min
    call iprint    ; Вывод наименьшего числа
    call quit       ; Завершение программы
```

2. Напишите программу, которая для введенных с клавиатуры значений x и a вычисляет значение заданной функции f(x) и выводит результат вычислений. Вид функции f(x) выбрать из таблицы 7.6 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу для значений x и a из 7.6. Я создал файл lab7-4.asm и написал в нем код для вычисления функции в соответствии с моим вариантом (рис. **??**).

20	$\begin{cases} x-a, & x \geq a \\ 5, & x < a \end{cases}$	(1;2)	(2;1)

и проверил его работу (рис. 4.3).

```
мпьютера/arch-pc/lab07 $ touch lab7-4.asm
kachelukhaev1@dk2n23 ~/work/study/2024-2025/Архитектура компьютера/arch-pc/la
b07 $ nasm -f elf lab7-4.asm
kachelukhaev1@dk2n23 ~/work/study/2024-2025/Архитектура компьютера/arch-pc/la
b07 $ ld -m elf_i386 -o lab7-4 lab7-4.o
kachelukhaev1@dk2n23 ~/work/study/2024-2025/Архитектура компьютера/arch-pc/la
b07 $ ./lab7-4
Введите значение x: 1
Введите значение a: 2
Результат: 5kachelukhaev1@dk2n23 ~/work/study/2024-2025/Архитектура компьютер
a/arch-pc/lab07 $ ./lab7-4
Введите значение x: 2
Введите значение a: 1
Результат: 1kachelukhaev1@dk2n23 ~/work/study/2024-2025/Архитектура компьютер
a/arch-pc/lab07 $
```

Рис. 4.3: 7

```
%include 'in_out.asm'   ; Подключение внешнего файла

SECTION .data
    msg_x:   db 'Введите значение x: ', 0
    msg_a:   db 'Введите значение a: ', 0
    msg_res: db 'Результат: ', 0

SECTION .bss
    x:       resd 4     ; Переменные для хранения x и a
    a:       resd 4

SECTION .text
    global _start

_start:
    ; ---- Ввод x ----
    mov eax, msg_x    ; Вывод сообщения "Введите значение x:"
    call sprint
    mov ecx, x        ; Адрес переменной x
    mov edx, 10       ; Максимальная длина ввода
    call sread
    mov eax, x
    call atoi         ; Преобразование строки в целое, результат в EAX
    mov [x], eax

    ; ---- Ввод a ----
    mov eax, msg_a    ; Вывод сообщения "Введите значение a:"
    call sprint
    mov ecx, a        ; Адрес переменной a
```

```
mov edx, 10       ; Максимальная длина ввода
call sread
mov eax, a
call atoi         ; Преобразование строки в целое, результат в EAX
mov [a], eax

; ---- Проверка условия x >= a ----
mov eax, [x]      ; Загрузка значения x
mov ebx, [a]      ; Загрузка значения a
cmp eax, ebx      ; Сравнение x и a
jge calc_x_minus_a  ; Если x >= a, переходим к вычислению x - a

; ---- Если x < a, то f(x) = 5 ----
mov eax, 5        ; Загрузка значения 5 в eax
jmp print_result   ; Переход к выводу результата

; ---- Вычисление x - a ----
calc_x_minus_a:
    mov eax, [x]   ; Загрузка значения x
    sub eax, [a]   ; Вычитание a из x (eax = x - a)

; ---- Вывод результата ----
print_result:
    mov edi, eax     ; Результат в edi
    mov eax, msg_res  ; Вывод сообщения "Результат: "
    call sprint
    mov eax, edi    ; Вывод значения результата
    call iprint
    call quit         ; Завершение программы
```

5 Выводы

В итоге я изучил команды условного и безусловного переходов и приобрел навыки написания программ с использованием переходов. познакомился с назначением и структурой файла листинга.

Список литературы