Отчет по лабораторной работе №7

Дисциплина: Архитектура компьютера

Краснова Камилла Геннадьевна

Содержание

# 1 Цель работы

Цель данной лабораторной работы - изучение команд условного и безусловного переходов, приобретение навыков написания программ с использованием переходов, знакомство с назначением и структурой файла листинга.

# 2 Задание

1. Реализация переходов в NASM
2. Изучение структуры файлы листинга
3. Выполнение заданий 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 Безусловный переход выполняется инструкцией jmp (от англ. jump – прыжок), которая включает в себя адрес перехода, куда следует передать управление: jmp Адрес перехода может быть либо меткой, либо адресом области памяти, в которую предварительно помещен указатель перехода. Кроме того, в качестве операнда можно использовать имя регистра, в таком случае переход будет осуществляться по адресу, хранящемуся в этом регистре. Как отмечалось выше, для условного перехода необходима проверка какого-либо условия. В ассемблере команды условного перехода вычисляют условие перехода анализируя флаги из регистра флагов. Флаг – это бит, принимающий значение 1 («флаг установлен»), если выполнено некоторое условие, и значение 0 («флаг сброшен») в противном случае. Флаги работают независимо друг от друга, и лишь для удобства они помещены в единый регистр — регистр флагов, отражающий текущее состояние процессора.

# 4 Выполнение лабораторной работы

## 4.1 Реализация переходов в NASM

С помощью утилиты mkdir создаю директорию, в которой буду создавать файлы с программами для лабораторной работы №7 Перехожу в созданный каталог с помощью утилиты cd и с помощью утилиты touch создаю файл lab7-1.asm.(рис. 1).

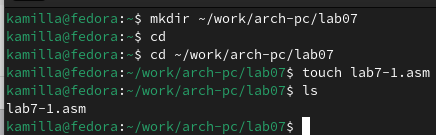


Рис. 1: Создание директории

Открываю созданный файл lab7-1.asm, вставляю в него программу с использованием инструкции jmp (рис. 2).

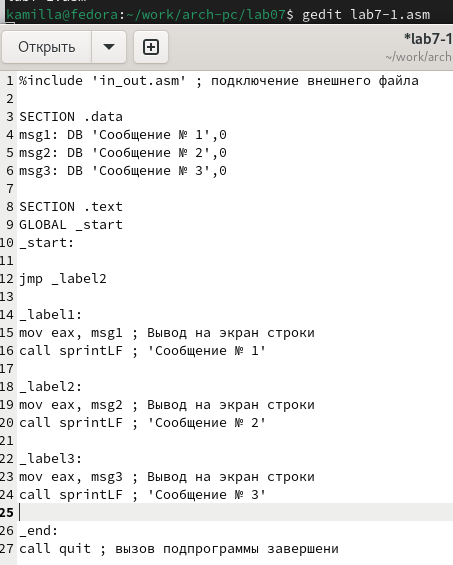


Рис. 2: Редактирование файла

Создаю исполняемый файл программы и запускаю его (рис. 3).

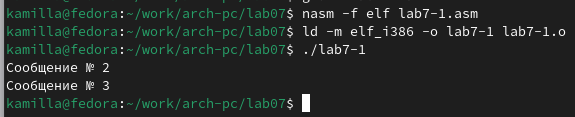
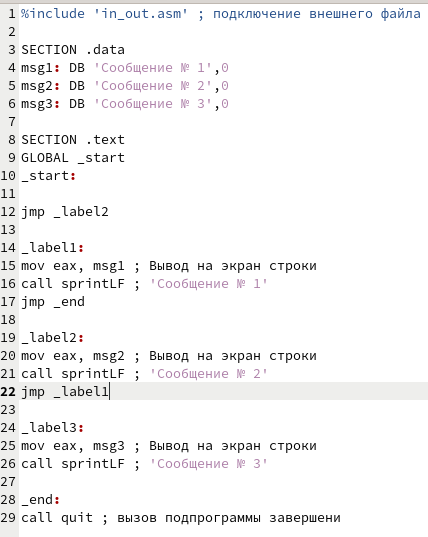


Рис. 3: Запуск исполняемого файла

Изменяю текст программы (рис. **¿fig:004?**).

{ #fig:006=4 width=70% }

Создаю новый исполняемый файл программы и запускаю его (рис. 4).

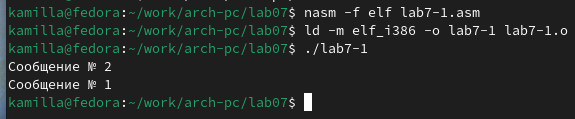


Рис. 4: Запуск исполняемого файла

Изменяю текст программы, чтобы она выводила сначала сообщение №3, затем сообщение №2 и затем сообщение №1 (рис. 5).

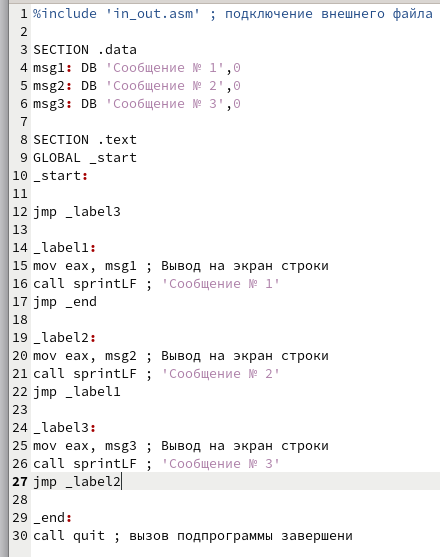


Рис. 5: Редактирование файла

Создаю новый исполняемый файл программы и запускаю его (рис. 6).

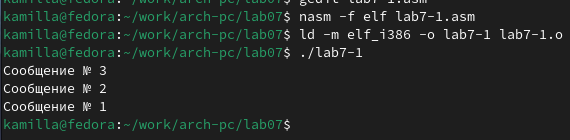


Рис. 6: Запуск исполняемого файла

Создаю файл lab7-2 в каталоге ~/work/arch-pc/lab07 (рис. 7).

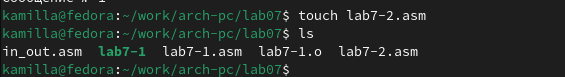


Рис. 7: Создание файла

Открываю созданный файл lab7-2.asm, вставляю в него программу, которая определяет и выводит на экран наибольшую из 3 целочисленных переменных: A,B и C. (рис. 8).

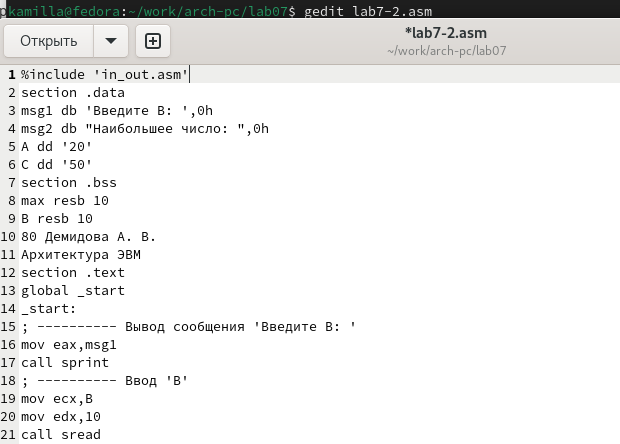


Рис. 8: Редактирование файла

Создаю и запускаю новый исполняемый файл (рис. 9). Проверяю работу для разных значений В. Программа работает верно.

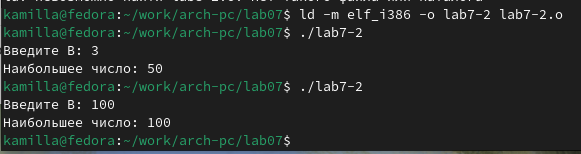


Рис. 9: Запуск исполняемого файла

## 4.2 Изучение структуры файла листинга

Создаю файл листинга для программы из файла lab7-2.asm, с помощью ключа -l и открываю файл листинга с помощью gedit (рис. 10).

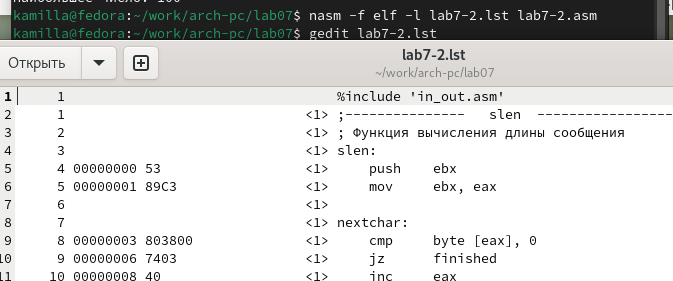


Рис. 10: Создание файла

Первое - номер строки файла листинга, важно понимать, что он может не совпадать с номером строки в файле с исходным текстом. Дальше идет адрес - смещение машинного кода от начала текущего сегмента. Следом идет машинный код, который представлчет собой ассемблированную исходную строку в виде шестнадцатиричной последовательности. И исходный текст программы - строка исходной программы вместе с комментариями.

Открываю файл с программой lab7-2.asm и удаляю один операнд (рис. 11).

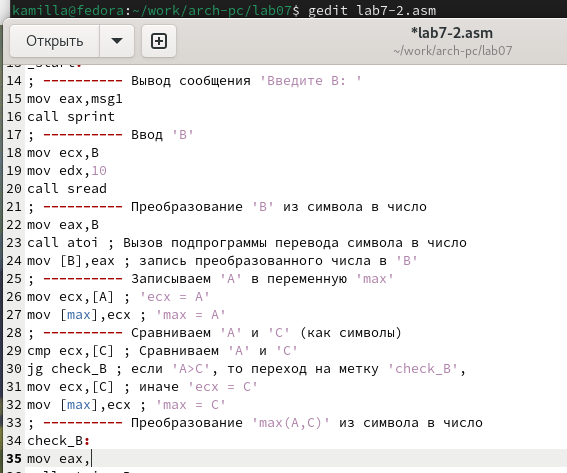


Рис. 11: Редактирование файла

В новом файле листинга показывает ошибку, которая возникла при попытке трансляции. Не создаются входные данные (рис. 12).

Рис. 12: Трансляция с получением файла листинга

Рис. 12: Трансляция с получением файла листинга

## 4.3 Выполнение заданий для самостоятельной работы

1. Создаю файл lab7-3.asm с помощью утилиты touch (рис. 13).

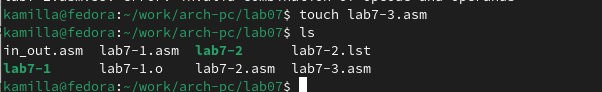


Рис. 13: Создание файла

Ввожу в созданный файл программу нахождения наименьшей из 3 целочисленных переменных 𝑎,𝑏 и с. (рис. 14).

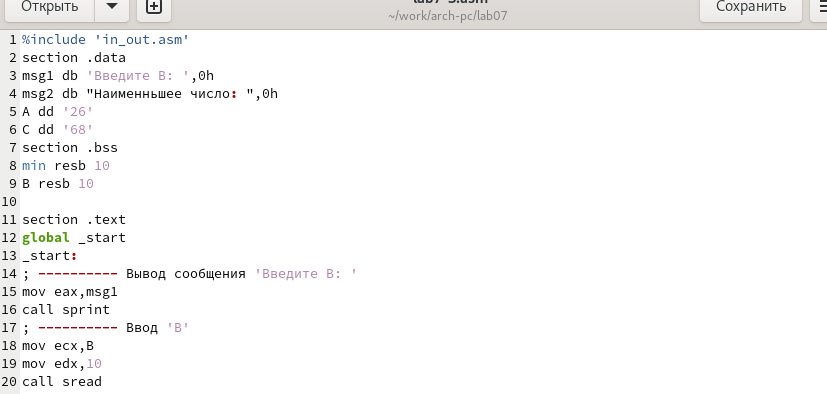


Рис. 14: Редакирование файла

Создаю и запускаю новый исполняемый файл. Мой вариант - 17, ввожу соответсвующие значения (рис. 15).

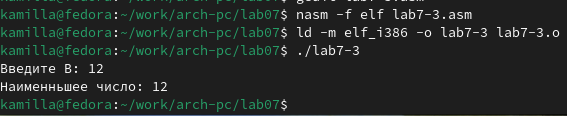


Рис. 15: Запуск исполняемого файла

1. Создаю файл lab7-4.asm с помощью утилиты touch (рис. 16).

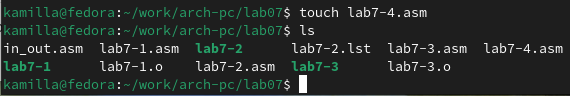


Рис. 16: Создание файла

Ввожу в созданный файл программу которая для введенных с клавиатуры значений 𝑥 и 𝑎 вычисляет значение заданной функции 𝑓(𝑥) и выводит результат вычислений (рис. 17).

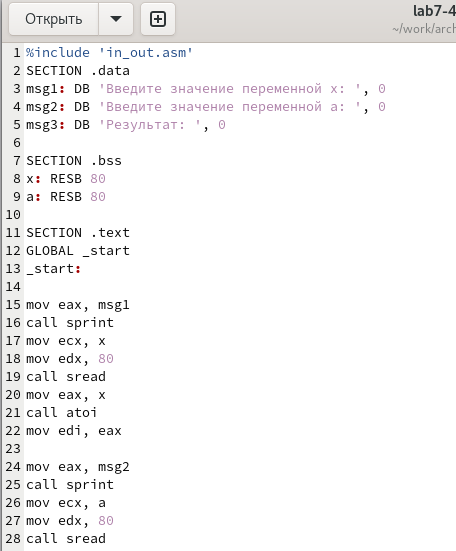


Рис. 17: Редактирование файла

Создаю и запускаю исполняемый файл (рис. 18). Проверила результат вручную, он сошелся с результатом программы.

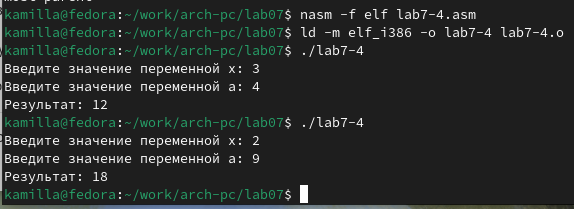


Рис. 18: Запуск исполняемого файла

**Листинг. Программа для нахождения наименьшей из 3 целочисленных переменных 𝑎,𝑏 и с.**

```%include ‘in\_out.asm’ section .data msg1 db ‘Введите B:’,0h msg2 db “Наименньшее число:”,0h A dd ‘26’ C dd ‘68’ section .bss min 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 ‘min’ mov ecx,[A] ; ‘ecx = A’ mov [min],ecx ; ‘min = A’ ; ———- Сравниваем ‘A’ и ‘С’ (как символы) cmp ecx,[C] ; Сравниваем ‘A’ и ‘С’ jl check\_B ; если ‘A>C’, то переход на метку ‘check\_B’, mov ecx,[C] ; иначе ‘ecx = C’ mov [min],ecx ; ‘min = C’ ; ———- Преобразование ‘min(A,C)’ из символа в число check\_B: mov eax,min call atoi ; Вызов подпрограммы перевода символа в число mov [min],eax ; запись преобразованного числа в min ; ———- Сравниваем ‘min(A,C)’ и ‘B’ (как числа) mov ecx,[min] cmp ecx,[B] ; Сравниваем ‘min(A,C)’ и ‘B’ jl fin ; если ‘min(A,C)>B’, то переход на ‘fin’, mov ecx,[B] ; иначе ‘ecx = B’ mov [min],ecx ; ———- Вывод результата fin: mov eax, msg2 call sprint ; Вывод сообщения ‘Наибольшее число:’ mov eax,[min] call iprintLF ; Вывод ‘max(A,B,C)’ call quit ; Выход

\*\*Листинг. Программа, которая для введенных с клавиатуры значений 𝑥 и 𝑎 вычисляет значение заданной функции 𝑓(𝑥) и выводит результат вычислений.\*\*  
  
```%include 'in\_out.asm'  
SECTION .data  
msg1: DB 'Введите значение переменной x: ', 0  
msg2: DB 'Введите значение переменной a: ', 0  
msg3: DB 'Результат: ', 0  
  
SECTION .bss  
x: RESB 80  
a: RESB 80  
  
SECTION .text  
GLOBAL \_start  
\_start:  
  
mov eax, msg1  
call sprint  
mov ecx, x  
mov edx, 80  
call sread  
mov eax, x  
call atoi  
mov edi, eax  
  
mov eax, msg2  
call sprint  
mov ecx, a  
mov edx, 80  
call sread  
mov eax, a  
call atoi  
mov esi, eax  
  
cmp esi, 8  
jl add\_values  
mov eax, esi  
mov ebx, edi  
mul ebx  
jmp print\_result  
  
add\_values:  
mov eax, esi  
add eax, 8  
  
print\_result:  
mov edi, eax  
mov eax, msg3  
call sprint  
mov eax, edi  
call iprintLF  
call quit

# 5 Выводы

При выполнении данной лабораторной работы я изучила команды условного и безусловного перехода, приобрела навыки написания программ с использованием переходов, познакомилась с назначением и структурой файла листинга.

# 6 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)