# Отчет по лабораторной работе №7
# Номер студенческого билета: 10322536339

Габдуллина Александра Булатовна

# 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 Задание

В качестве ответа на лабораторную работу в ТУИС необходимо загрузить :

1.Ссылка на репозиторий (https://github.com/) 2.Следует загрузить отдельными файлами (не общим архивом): 3.отчёт в markdown; 4.отчёт в docx (сделанный из markdown); 5.отчёт в pdf (сделанный из markdown); 6.архив с исходными материалами markdown (текстовые файлы, скриншоты и т. д.) 7.Файл(ы) с исходным текстом программ, написанных в ходе выполнения работы (в формате .asm).

# 3 Выполнение лабораторной работы

Создаём каталог для програм лабораторной работы № 7, переодим в него и со- здаём файл lab7-1.asm:

```
gabdullinaab@ubuntu:~$ mkdir ~/work/arch-pc/lab07
gabdullinaab@ubuntu:~$ cd ~/work/arch-pc/lab07
gabdullinaab@ubuntu:~/work/arch-pc/lab07$ touch lab7-1.asm
```

Рисунок 3.1: Создание каталога, переход в каталог и создание файла lab7-1.asm

Введим в файл lab7-1.asm текст программы из листинга 7.1.

Open | *lab7-1.asm ~/work/arch-pc/lab07 | Save

```
1 SECTION .data
2 msg1: DB 'Сообщение № 1',0
3 msg2: DB 'Сообщение № 2',0
4 msg3: DB 'Сообщение № 3',0
5 SECTION .text
6 GLOBAL _start
7 _start:
8 jmp _label2
9 _label1:
10 mov eax, msg1 ; Вывод на экран строки
11 call sprintLF ; 'Сообщение № 1'
12 _label2:
13 mov eax, msg2 ; Вывод на экран строки
14 call sprintLF ; 'Сообщение № 2'
15 _label3:
16 mov eax, msg3 ; Вывод на экран строки
17 call sprintLF ; 'Сообщение № 3'
18 _end:
19 call quit ; вызов подпрограммы завершения
```

Рисунок 3.2: Вводим текст программы в файл lab7-1.asm

Создаём исполняемый файл и запускаем его.

```
gabdullinaab@ubuntu:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
gabdullinaab@ubuntu:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
gabdullinaab@ubuntu:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
gabdullinaab@ubuntu:~/work/arch-pc/lab07$
```

Рисунок 3.3: Создание файла и запуск

Изменяем текст программы в соответствии с листингом 7.2.

Open | *lab7-1.asm ~/work/arch-pc/lab07 | Save

```
%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
```

Рисунок 3.4: Изменение текста программы

Создаём исполняемый файл и проверяем его работу.

```
gabdullinaab@ubuntu:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
gabdullinaab@ubuntu:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
gabdullinaab@ubuntu:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 1
gabdullinaab@ubuntu:~/work/arch-pc/lab07$
```

Рисунок 3.5: Создание файла и проверка его работы

Снова изменяем текст программы, чтобы вывод программы был следующим: Сообщение № 3 Сообщение № 2 Сообщение № 1

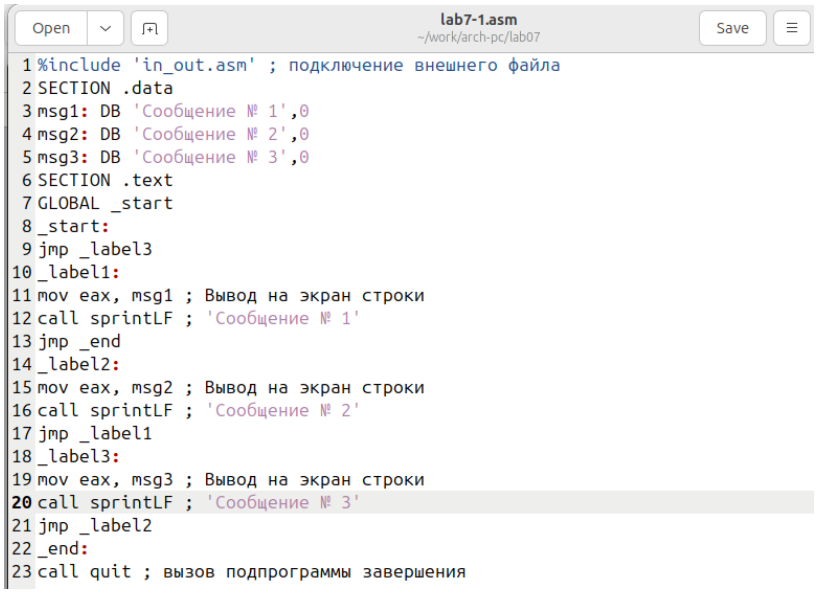


Рисунок 3.6: Изменяем текст программы

Создаём исполняемый файл и проверяем его работу.

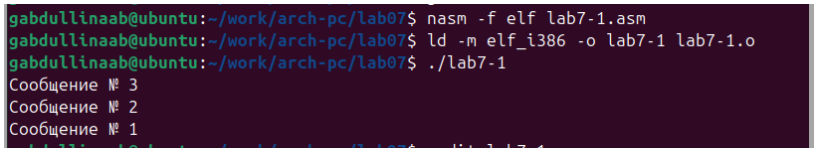


Рисунок 3.7: Создание файла и проверка его работы

Создаём файл lab7-2.asm в каталоге ~/work/arch-pc/lab07.

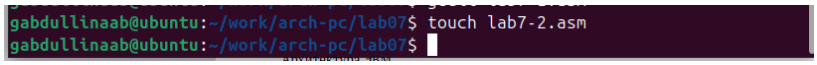


Рисунок 3.8: Создание файла lab7-2.asm

Вводим текст программы из листинга 7.3 в файл в lab7-2.asm, запускаем файл и проверяем его работу для различных значений B.

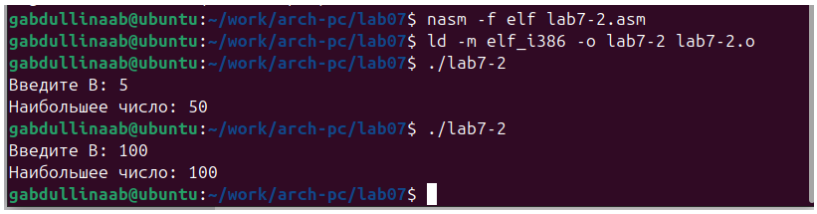


Рисунок 3.9: Создание, запуск и проверка файла lab7-2.asm для значений 5 и 100

Создаём файл листинга для программы из файла lab7-2.asm и с помощью текстового редактора mcedit открываем его.

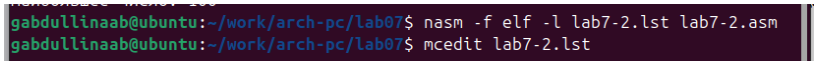


Рисунок 3.10: Создание файла листинга для программы из файла и открывание его

Ознакомимся с его форматом и содержимым.Объяснение содержимого трёх строк файла листинга по выбору:

Строка 5 push ebx - инструкция сохраняет значение регистра EBX в стек Машинный код: 53 - шестнадцатеричное представление инструкции PUSH EBX Адрес: 00000000 - относительный адрес в памяти, где будет расположена эта инструкция <1> - номер исходного файла (основной файл)

Строка 8 cmp byte [eax], 0 - сравнивает байт по адресу в EAX с нулём Машинный код: 803800 - состоит из: 80 - код операции CMP 38 - модификатор для сравнения байта в памяти 00 - значение для сравнения (ноль) Адрес: 00000003 - позиция в памяти

Проверяем, не достигли ли мы конца строки (строки в C заканчиваются нулевым байтом)

Строка 9 jz finished - переход к метке finished, если предыдущее сравнение было равно 0 Машинный код: 7403 - 74 = код JZ, 03 = смещение на 3 байта вперёд Адрес: 00000006

Если найден,нулевой байт(конец строки), выходим из цикла .

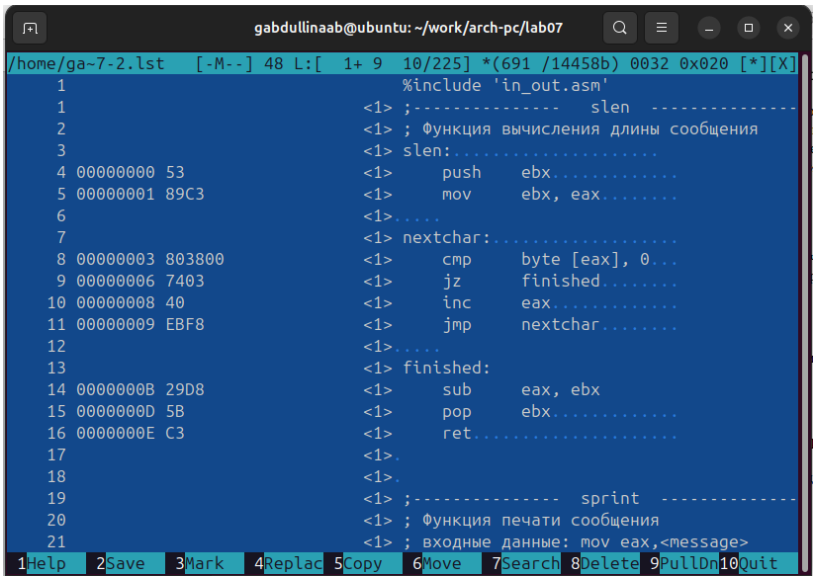


Рисунок 3.11: Файл lab7-2.lst

В файле lab7-2.asm в строке 31 нужно удалить второй операнд:

Было:

31. mov [max],ecx ; „max = C“

Стало:

31. mov [max] ; „max = C“

Открываем файл с программой lab7-2.asm и в 31 строке удаляем один операнд.

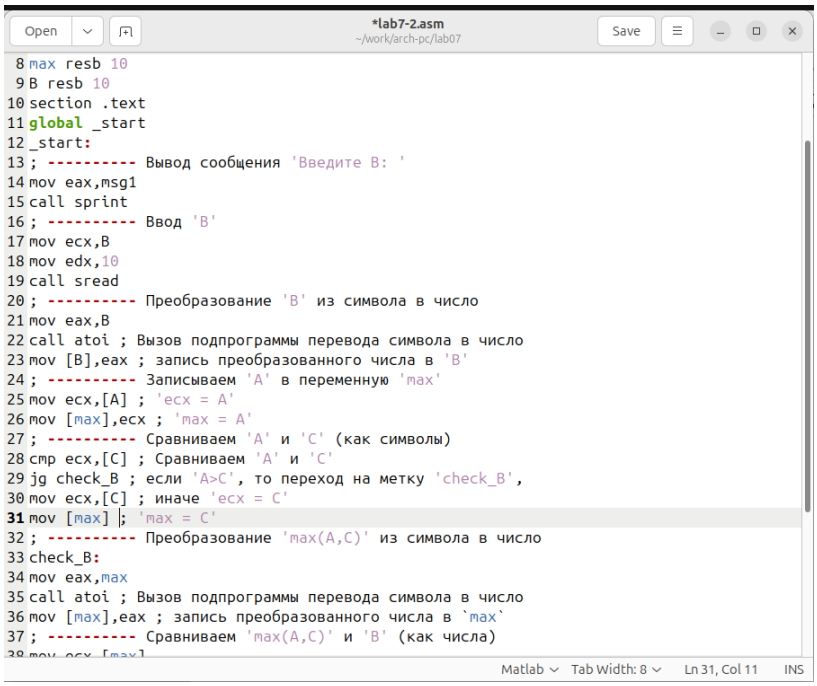


Рисунок 3.12: Удаление операнда в 31 строке в файле

Выполняем трансляцию с получением файла листинга.

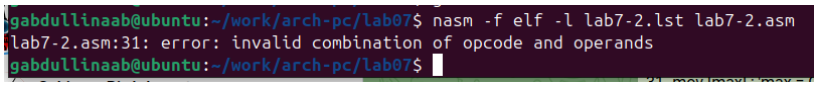


Рисунок 3.13: Выполняем транляцию

С помощью команды ls проверим какие файлы у нас создались.

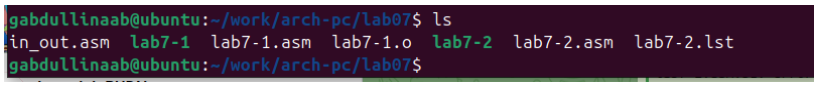


Рисунок 3.14: Проверяем, какие файлы у нас создались

У нас создались:

1) lab7-2.lst - листинг (пустой или с ошибкой)
2) Объектный файл lab7-2.o НЕ создался - потому что трансляция завершилась с ошибкой

Посмотрим, что у нас добавилось в листинге.

```
31          ****************        error: invalid combination of opcode and operands
```

Рисунок 3.15: Строка 31 в листинге

#Задание для самостоятельной работы

Напишем программу нахождения наименьшей из 3 целочисленных переменных 95,2,61 (вариант 20) .

*min3.asm
~/work/arch-pc/lab07

```
1 section .data
2     a dd 95
3     b dd 2
4     c dd 61
5     msg db "Наименьшее число: ", 0
6     newline db 10, 0
7
8 section .text
9     global _start
10    %include 'in_out.asm'  ; Подключаем файл с функциями ввода-вывода
11
12 _start:
13    ; Сначала предполагаем, что a - наименьшее
14    mov eax, [a]
15    mov ebx, [b]
16    mov ecx, [c]
17
18    ; Сравниваем a и b
19    cmp eax, ebx
20    jle compare_c  ; Если a <= b, переходим к сравнению с c
21    mov eax, ebx   ; Иначе наименьшее = b
22
23 compare_c:
24    ; Сравниваем текущее наименьшее (eax) с c
25    cmp eax, ecx
26    jle print_result  ; Если текущее <= c, переходим к выводу
27    mov eax, ecx      ; Иначе наименьшее = c
28
29 print_result:
30    ; Выводим сообщение
31    push eax          ; Сохраняем результат
```

Рисунок 3.16: Текст программы задания 1 из самостоятельной работы

Создаём исполняемый файл и проверяем его работу.

```
gabdullinaab@ubuntu:~/work/arch-pc/lab07$ nasm -f elf min3.asm
gabdullinaab@ubuntu:~/work/arch-pc/lab07$ ld -m elf_i386 min3.o -o min3
gabdullinaab@ubuntu:~/work/arch-pc/lab07$ ./min3
Наименьшее число: 2
gabdullinaab@ubuntu:~/work/arch-pc/lab07$
```

Рисунок 3.17: Создание файла и проверка его работы

Напишем программу, которая для введенных с клавиатуры значений и вычисляет значение заданной функции ( ) и выводит результат вычислений.Вид функции ( ) выбрала из таблицы 7.6. (вариант 20)

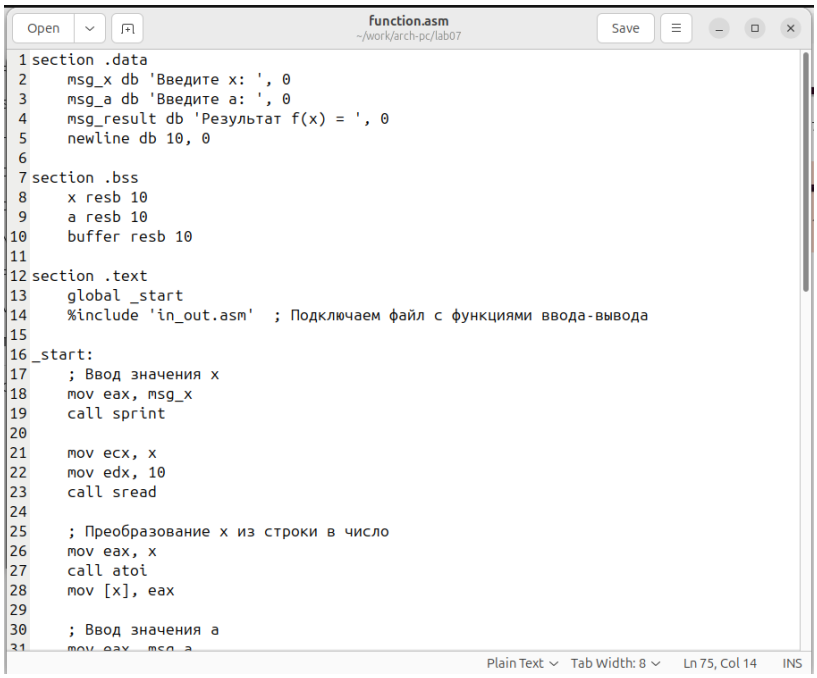


Рисунок 3.18: Текст программы задания 2 из самостоятельной работы

Создаём исполняемый файл и проверяем его работу для значений 1=1, x2 =2, a1= 2, a2= 1 из 7.6.

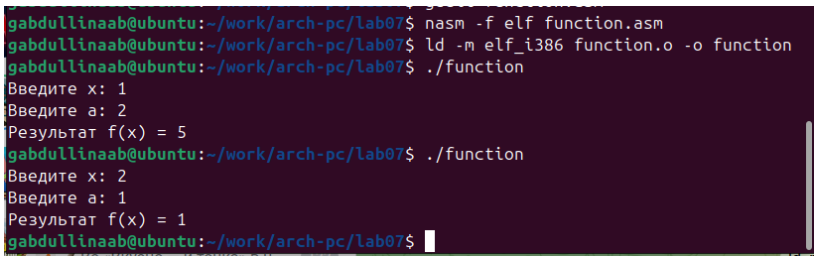


Рисунок 3.19: Создание файла и проверка его работы

# 4 Выводы

В ходе лабораторной работы я изучила команды условного и безусловного переходов, приобрела навыки написания программ с использованием переходов, познакомилась с назначением и структурой файла листинга.