Отчёт по лабораторной работе №7

Дисциплина: Архитектура компьютера

София Андреевна Кудякова

Содержание

# 1 Цель работы

Цель данной лабораторной работы - изучить команды условного и безусловного переходов, научиться писать программы с использованием переходов, а также ознакомиться с назначением и структурой файла листинга.

# 2 Задания

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: 1. Условный переход – выполнение или не выполнение перехода в определенную точку программы в зависимости от проверки условия. 2. 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. 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. В следующей таблице указано положение битовых флагов в регистре флагов. Флаги состояния (биты 0, 2, 4, 6, 7 и 11) отражают результат выполнения арифметических инструкций, таких как ADD, SUB, MUL, DIV.Инструкция cmp является одной из инструкций, которая позволяет сравнить операнды и выставляет флаги в зависимости от результата сравнения. Инструкция cmp является командой сравнения двух операндов и имеет такой же формат, как и команда вычитания: cmp , Команда cmp, так же как и команда вычитания, выполняет вычитание - , но результат вычитания никуда не записывается и единственным результатом команды сравнения является формирование флагов. Примеры:

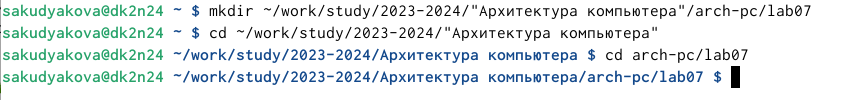
cmp ax,'4' ; сравнение регистра ax с символом 4  
cmp ax,4 ; сравнение регистра ax со значением 4  
cmp al,cl ; сравнение регистров al и cl  
cmp [buf],ax ; сравнение переменной buf с регистром ax

Команда условного перехода имеет вид: j label Мнемоника перехода связана со значением анализируемых флагов или со способом формирования этих флагов. Листинг (в рамках понятийного аппарата NASM) — это один из выходных файлов, создаваемых транслятором. Он имеет текстовый вид и нужен при отладке программы, так как кроме строк самой программы он содержит дополнительную информацию.

# 4 Выполнение лабораторной работы

## 4.1 Реализация переходов в NASM

Ввожу команду mkdir, с помощью которой создаю директорию, в которой буду создавать файлы. Перехожу в нее. (рис. ??).



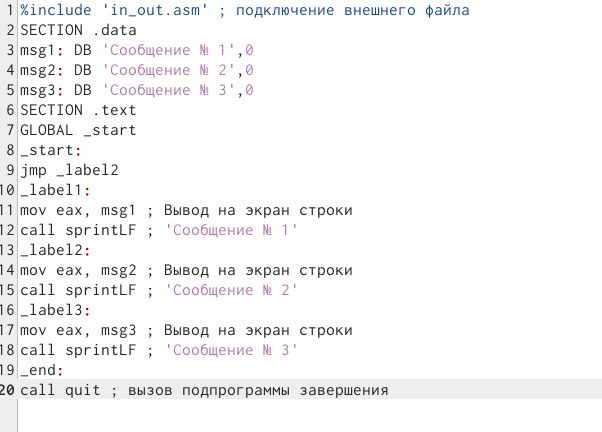
Создание директории

С помощью команды touch создаю файл lab7-1.asm. (рис. ??).

Создание файла

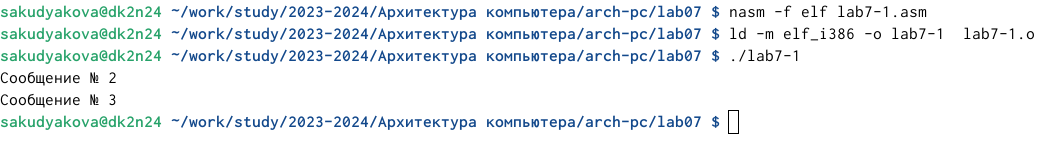
Создание файла

Открываю созданный файл в редакторе и вставляю в него программу с использованием инструкции jmp. (рис. ??).



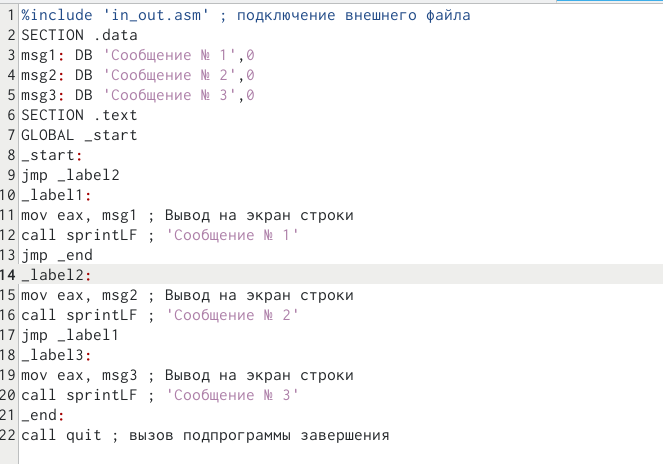
Редактирование файла

Создаю исполняемый файл и запускаю его. Программа сработала корректно. (рис. ??).



Запуск программы файла

Инструкция jmp позволяет осуществлять переходы не только вперед но и назад. Изменяю программу таким образом, чтобы она выводила сначала ‘Сообщение № 2’, потом ‘Сообщение № 1’ и завершала работу. Для этого в текст программы после вывода сообщения № 2 добавим инструкцию jmp с меткой \_label1 (т.е. переход к инструкциям вывода сообщения № 1) и после вывода сообщения № 1 добавим инструкцию jmp с меткой \_end (т.е. переход к инструкции call quit). (рис. ??).



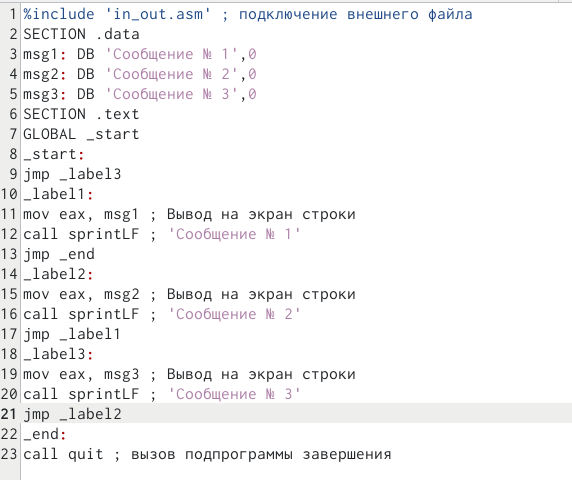
Редактирование файла

Создаю исполняемый файл и запускаю его. Программа отработала верно.(рис. ??).



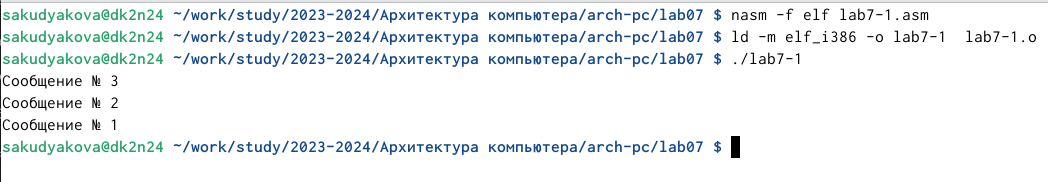
Запуск программы файла

Далее изменяю текст программы, добавив и изменив инструкции jmp. (рис. ??).



Редактирование файла

Создаю исполняемый файл и запускаю его. Вывод верный, значит программа отработала корректно. (рис. ??).



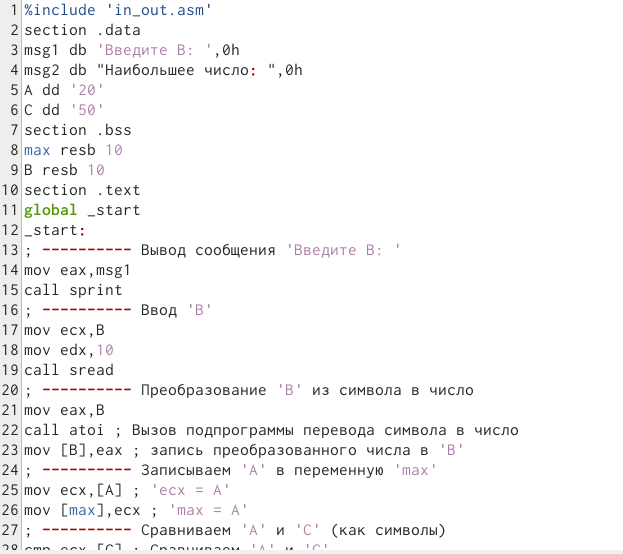
Исполнение файла

Создаю новый файл lab7-2.asm в каталоге ~/work/arch-pc/lab06. (рис. ??).

Создание файла

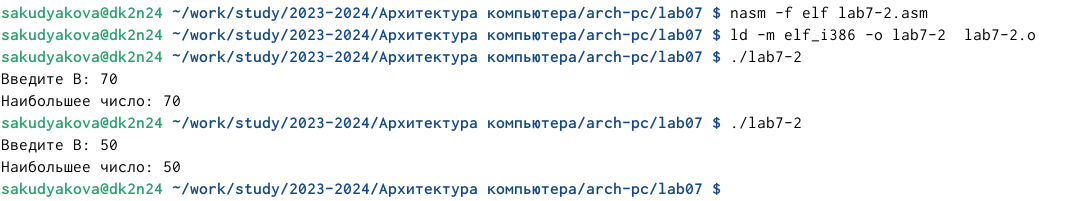
Создание файла

Ввожу в файл текст ррограмма, которая определяет и выводит на экран наибольшую из 3 целочисленных переменных: A,B и C. (рис. ??).



Ввод программы в файл

Создаю исполняемый файл и дважды запускаю его, чтобы проверить корректность работы программы. (рис. ??).



Исполнение файла

Программа отработала верно

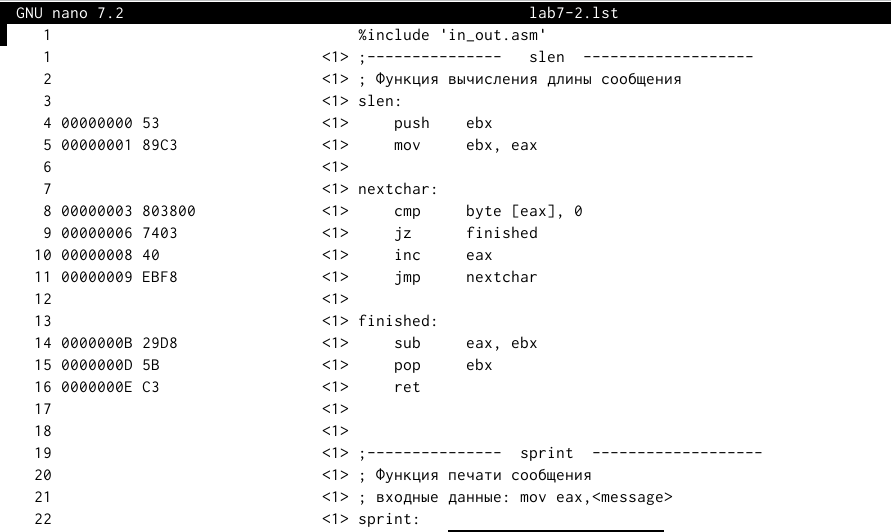
## 4.2 Изучение структуры файлы листинга

Создаю файл листинга для программы из файла lab7-2.asm. (рис. ??).

Создание файла листинга

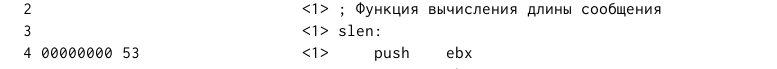
Создание файла листинга

Открываю файл листинга lab7-2.lst с помощью текстового редактора.(рис. ??).



Открытие файла

Для объяснения я выбрала строки 2; 3; 4. (рис. ??).



Выбанные строки

Строка “2” - номер строки кода; ” ; Функция вычисление длины сообщения” - комментарий к коду, не имеет адреса и машинного кода. Строка “3” - номер строки кода; “slen” - название функции, не имеет адреса и машинного кода. Строка “4” - номер строки кода; “00000000” - адрес строки; “53” - машинный код; “push ebx” - исходный текст программы, push помещает операнд ebx в стек.

Открываю файл с программой lab7-2.asm и в инструкции с двумя операндами удаляю один операнд(рис. ??).

Удаление операнда

Удаление операнда

Выполняю трансляцию с получением файла листинга. (рис. ??).

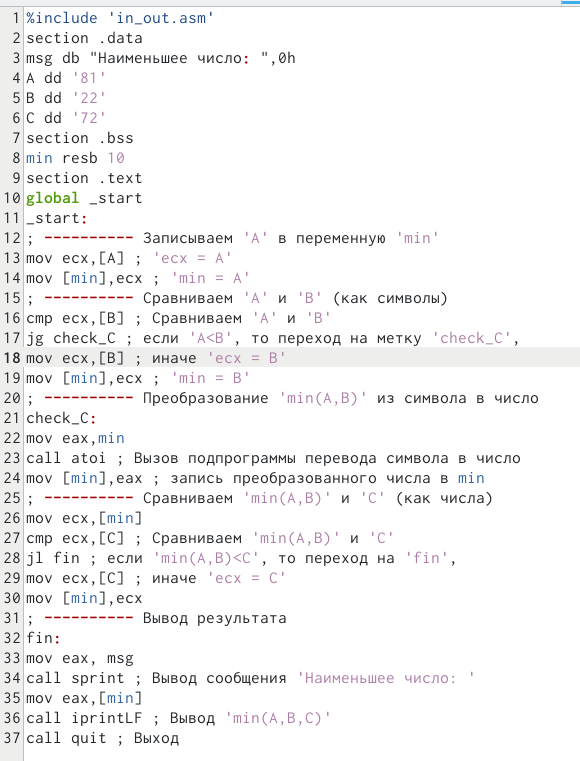
Трансляция с получением файла листинга

Трансляция с получением файла листинга

В этом случае не создаются никакие выходные файлы из-за ошибки: инструкция mov не может работать, имея только один операнд. Поэтому код не работает.

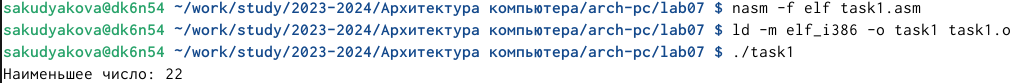
## 4.3 Выполнение заданий для самостоятельной работы

1. Пишу программу нахождения наименьшей из 3 целочисленных переменных a, b и c. Значения переменных выбрать из табл. 7.5 в соответствии с вариантом, полученным при выполнении лабораторной работы № 6. Мой вариант - 14. Мои значения - 81,22,72 (рис. ??).



Написание программы

Создаю исполняемый файл и запускаю его. (рис. ??).



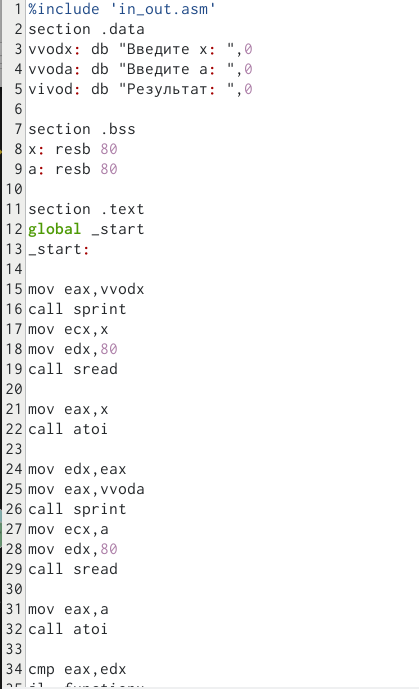
Исполнение файла

Программа отработала верно.

**Листинг 4.2.1. Программа, которая определяет и выводит на экран наименьшую из 3 целочисленных переменных: A,B и C**

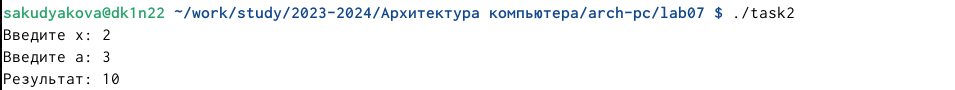
%include 'in\_out.asm'  
  
section .data  
msg db "Наименьшее число: ",0h  
A dd '81'  
B dd '22'  
C dd '72'  
section .bss  
min resb 10  
section .text  
global \_start  
\_start:  
; ---------- Записываем 'A' в переменную 'min'  
mov ecx,[A] ; 'ecx = A'  
mov [min],ecx ; 'min = A'  
; ---------- Сравниваем 'A' и 'B' (как символы)  
cmp ecx,[B] ; Сравниваем 'A' и 'B'  
jg check\_C ; если 'A<B', то переход на метку 'check\_C',  
mov ecx,[B] ; иначе 'ecx = B'  
mov [min],ecx ; 'min = B'  
; ---------- Преобразование 'min(A,B)' из символа в число  
check\_C:  
mov eax,min  
call atoi ; Вызов подпрограммы перевода символа в число  
mov [min],eax ; запись преобразованного числа в min  
; ---------- Сравниваем 'min(A,B)' и 'C' (как числа)  
mov ecx,[min]  
cmp ecx,[C] ; Сравниваем 'min(A,B)' и 'C'  
jl fin ; если 'min(A,B)<C', то переход на 'fin',  
mov ecx,[C] ; иначе 'ecx = C'  
mov [min],ecx  
; ---------- Вывод результата  
fin:  
mov eax, msg  
call sprint ; Вывод сообщения 'Наименьшее число: '  
mov eax,[min]  
call iprintLF ; Вывод 'min(A,B,C)'  
call quit ; Выход

1. Пишу программу, которая для введенных с клавиатуры значений x и a вычисляет значение заданной функции f(x) и выводит результат вычислений. Вид функции f(x) выбираю в соответствие с вариантом, полученным при выполнении лабораторной работы №6. Мой вариант 14, выбираю выражение для f(x) под номером 14. (рис. ??).



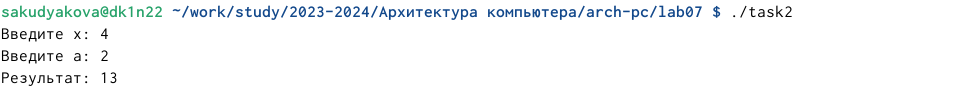
Написание программы

Создаю исполняемый файл и запускаю его. Ввожу сначала значения 2 и 3 (рис. ??).



Запуск исполняемого файла

Затем ввожу значения 4 и 2. (рис. ??).



Запуск исполняемого файла

Программа рабоает верно

**Листинг 4.2.2. Программа для вычисления значения заданной функции**

%include 'in\_out.asm'  
section .data  
vvodx: db "Введите x: ",0  
vvoda: db "Введите а: ",0  
vivod: db "Результат: ",0  
  
section .bss  
x: resb 80  
a: resb 80  
  
section .text  
global \_start  
\_start:  
  
mov eax,vvodx  
call sprint  
mov ecx,x  
mov edx,80  
call sread  
  
mov eax,x  
call atoi  
  
mov edx,eax  
mov eax,vvoda  
call sprint  
mov ecx,a  
mov edx,80  
call sread  
  
mov eax,a  
call atoi  
  
cmp eax,edx  
jl \_functionx  
jmp \_functiona  
  
\_functiona:  
mov edx,3  
mul edx  
add eax,1  
jmp \_end  
  
\_functionx:  
mov edx,3  
mul edx  
add eax,1  
jmp \_end  
  
\_end:  
mov ecx,eax  
mov eax,vivod  
call sprint  
mov eax,ecx  
call iprintLF  
call quit

# 5 Выводы

В ходе данной лабораторной работы я изучила команды условного и безусловного переходов, научилась писать программы с использованием переходов, а также ознакомилась с назначением и структурой файла листинга.

# Список литературы

[Архитектура ЭВМ](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)