Отчёт по лабораторной работе №9

Дисциплина: Архитектура компьютера

Кузьмина Мария Константиновна

Содержание

# 1 Цель работы

Приобретение навыков написания программ с использованием подпрограмм. Знакомство с методами отладки при помощи GDB и его основными возможностями.

# 2 Задание

1. Реализация подпрограмм в NASM
2. Отладка программ с помощью GDB
3. Выполнение заданий для самостоятельной работы.

# 3 Выполнение лабораторной работы

## 3.1 Реализация подпрограмм в NASM

С помощью mkdir создаем директорию для создания файлов лабораторной работы, переходим в созданный каталог(рис. 1):

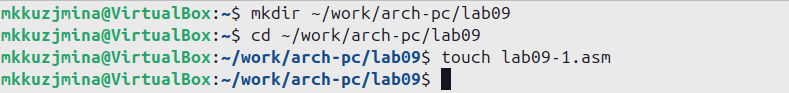


Рис. 1: снимок экрана

Копируем в файл код из листинга и запускаем его, данная программа выполняет вычисление функции (рис. 2):

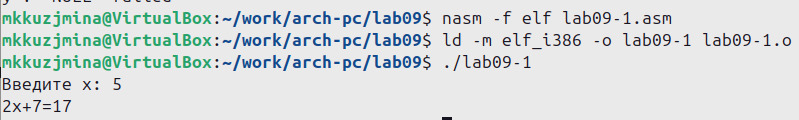


Рис. 2: снимок экрана

Изменяем текст программы, добавив подпрограмму \_subcalcul в подпрограмму \_calcul, для вычисления выражения 𝑓(𝑔(𝑥)), где 𝑥 вводится с клавиатуры, 𝑓(𝑥) = 2𝑥 + 7, 𝑔(𝑥) = 3𝑥 − 1. Т.е. 𝑥 передается в подпрограмму \_calcul из нее в подпрограмму \_subcalcul, где вычисляется выражение 𝑔(𝑥), результат возвращается в \_calcul и вычисляется выражение 𝑓(𝑔(𝑥)). (рис. 3)

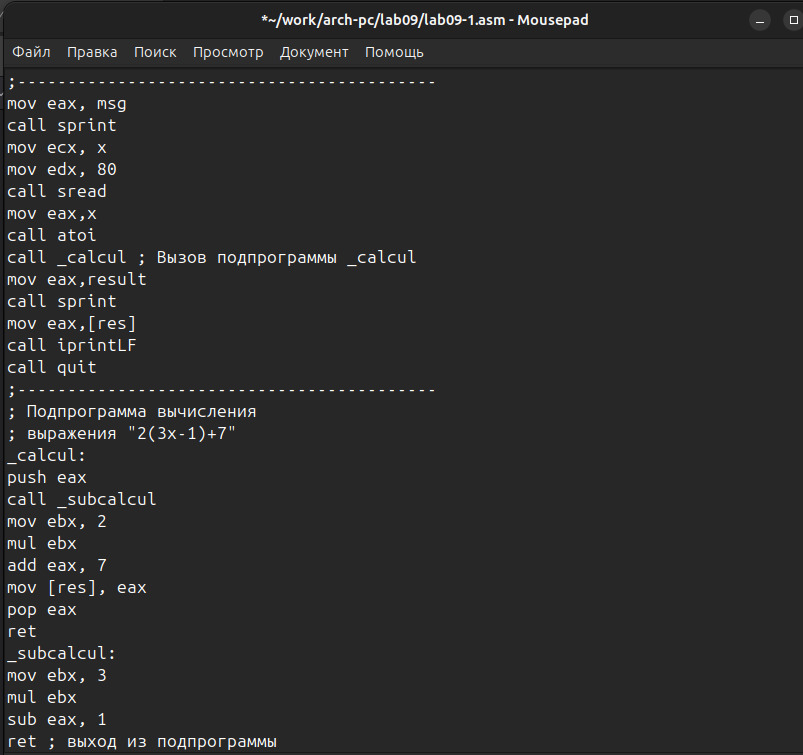


Рис. 3: снимок экрана

(рис. 4):

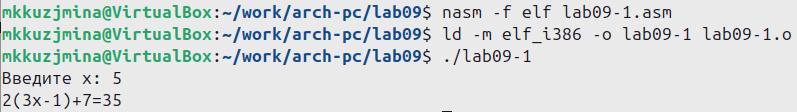


Рис. 4: снимок экрана

%include 'in\_out.asm'  
SECTION .data  
msg: DB 'Введите x: ',0  
result: DB '2(3x-1)+7=',0  
SECTION .bss  
x: RESB 80  
res: RESB 80  
SECTION .text  
GLOBAL \_start  
\_start:  
;------------------------------------------  
; Основная программа  
;------------------------------------------  
mov eax, msg  
call sprint  
mov ecx, x  
mov edx, 80  
call sread  
mov eax,x  
call atoi  
call \_calcul ; Вызов подпрограммы \_calcul  
mov eax,result  
call sprint  
mov eax,[res]  
call iprintLF  
call quit  
;------------------------------------------  
; Подпрограмма вычисления  
; выражения "2(3x-1)+7"  
\_calcul:  
push eax  
call \_subcalcul  
mov ebx, 2  
mul ebx  
add eax, 7  
mov [res], eax  
pop eax  
ret  
\_subcalcul:  
mov ebx, 3  
mul ebx  
sub eax, 1  
ret ; выход из подпрограммы

## 3.2 Отладка программ с помощью GDB

Создаем файл lab09-2.asm с текстом программы, в созданный файл копируем программу второго листинга, транслируем с созданием файла листинга и отладки, компонуем и запускаем в отладчике (рис. 5):

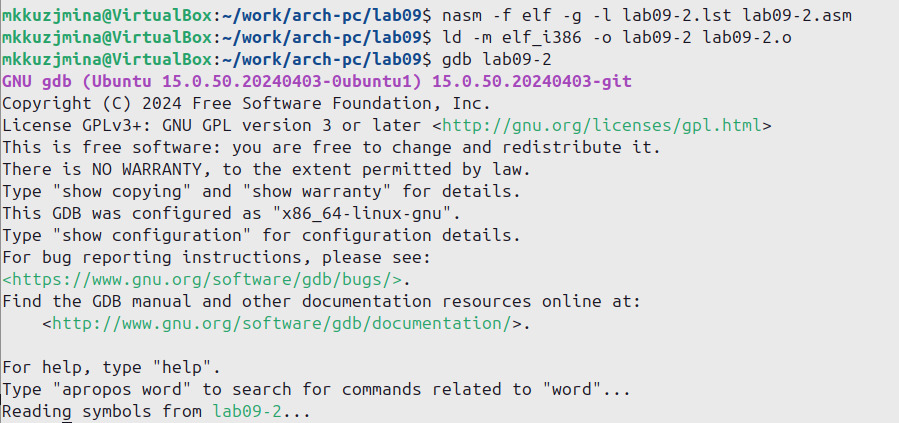


Рис. 5: снимок экрана

Проверяем работу программы, запустив ее в оболочке GDB с помощью команды run (рис. 6):

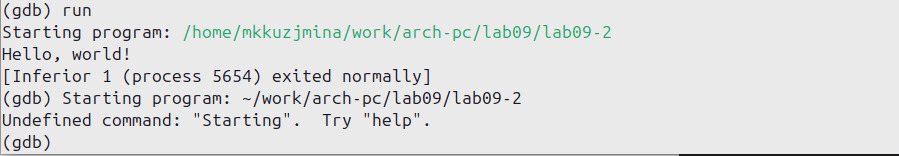


Рис. 6: снимок экрана

Для более подробного анализа программы устанавливаем брейкпоинт на метку \_start, с которой начинается выполнение любой ассемблерной программы, и запускаем её (рис. 7):

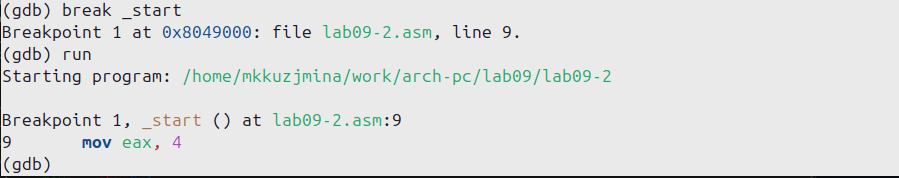


Рис. 7: снимок экрана

Смотрим дисассимилированный код программы с помощью команды disassemble начиная с метки \_start (рис. 8):



Рис. 8: снимок экрана

Переключаемся на отображение команд с Intel’овским синтаксисом, введя команду set disassembly-flavor intel (рис. 9):



Рис. 9: снимок экрана

Включаем режим псевдографики для более удобного анализа программы (рис. 10):

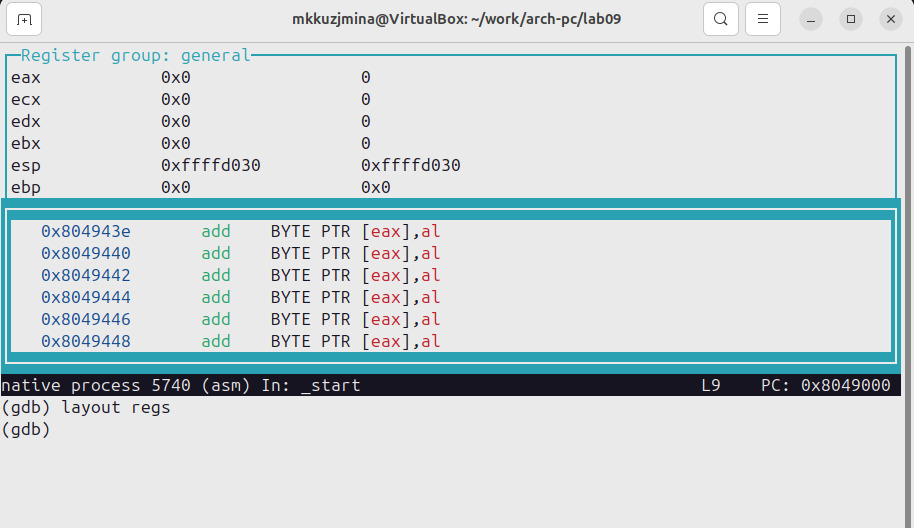


Рис. 10: снимок экрана

Установливаем точку останова (рис. 11):

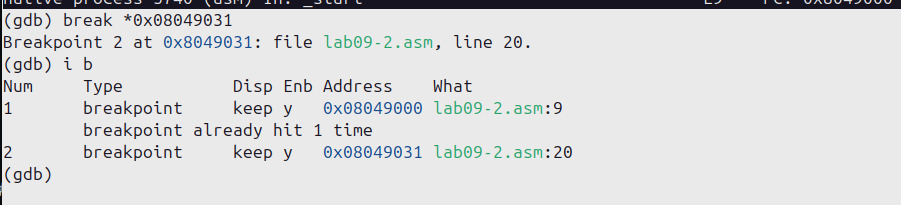


Рис. 11: снимок экрана

Смотрим содержимое регистров с помощью команды info registers (рис. 12):

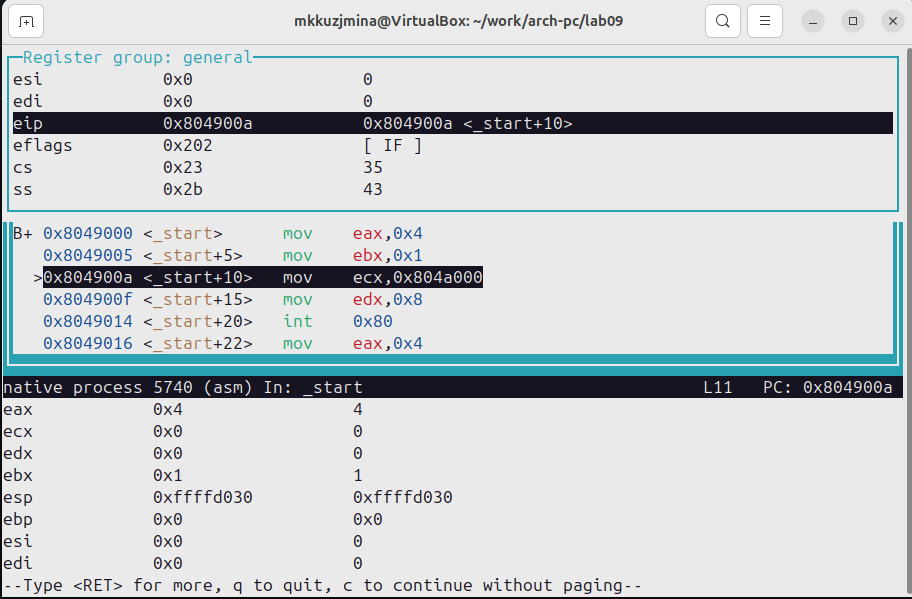


Рис. 12: снимок экрана

Смотрим значение переменной msg1 по имени (рис. 13):

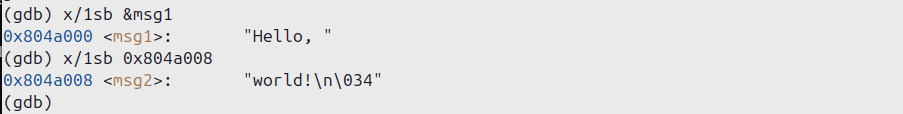


Рис. 13: снимок экрана

Изменяем первый символ переменной msg1 (рис. 14):

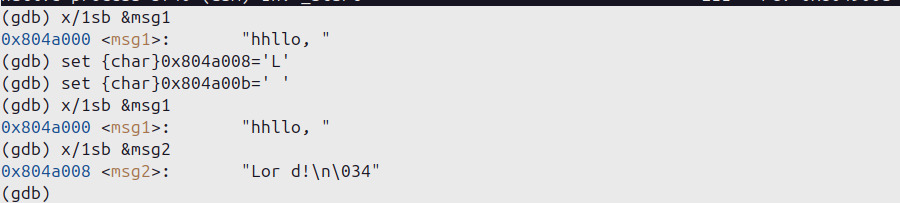


Рис. 14: снимок экрана

Выводим в различных форматах значение регистра edx (рис. 15):

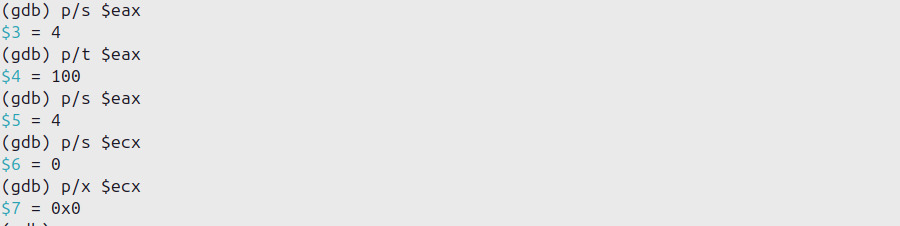


Рис. 15: снимок экрана

С помощью команды set меняем содержимое регистра ebx (рис. 16):

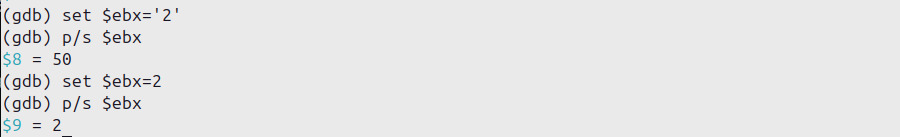


Рис. 16: снимок экрана

Копируем файл lab8-2.asm, созданный при выполнении лабораторной работы №8,с программой выводящей на экран аргументы командной строки (Листинг 8.2) в файл с именем lab09-3.asm (рис. 17):

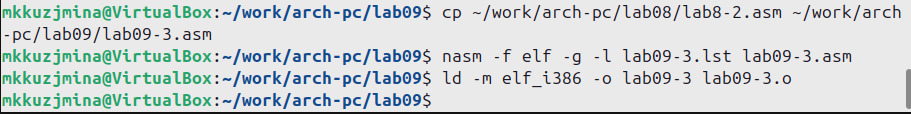


Рис. 17: снимок экрана

Для загрузки в gdb программы с аргументами используем ключ –args. Загружаем исполняемый файл в отладчик, указав аргументы. Установливаем точку останова перед первой инструкцией в программе и запускаем ее (рис. 18):

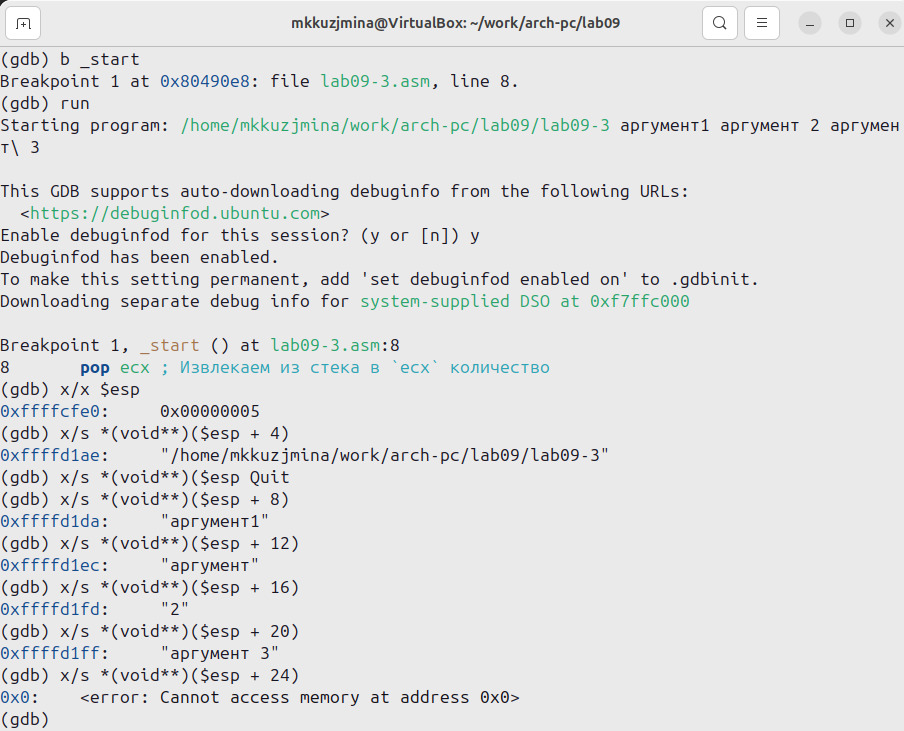


Рис. 18: снимок экрана

## 3.3 Выполнение заданий для самостоятельной работы

Преобразовываем программу из лабораторной работы №8 (Задание №1 для самостоятельной работы), реализовав вычисление значения функции 𝑓(𝑥) как подпрограмму (рис. 19):

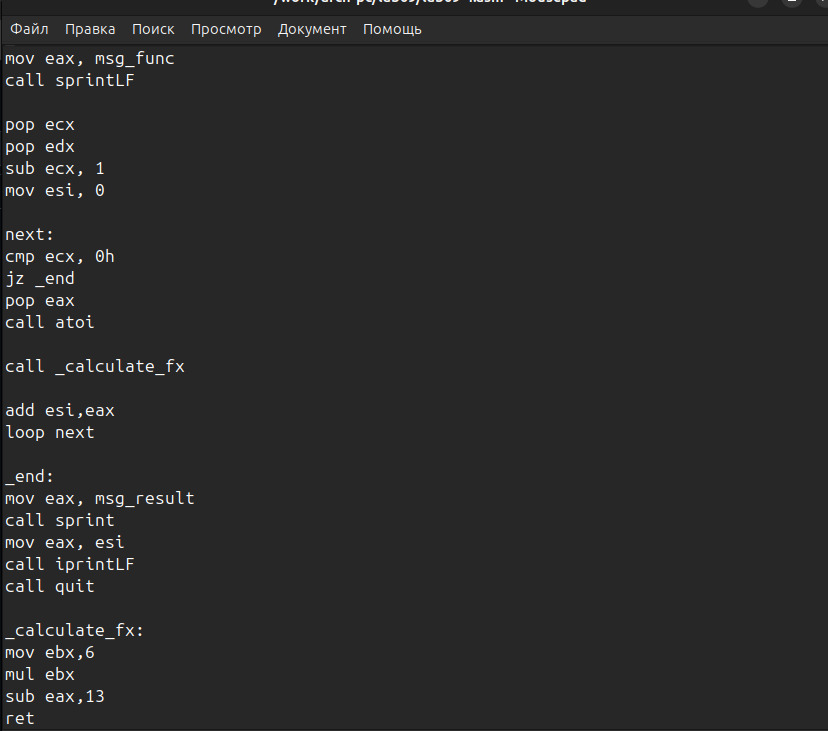


Рис. 19: снимок экрана

%include 'in\_out.asm'  
  
SECTION .data  
msg\_func db "Функция: f(x) = 6x + 13", 0  
msg\_result db "Результат: ", 0  
  
SECTION .text  
GLOBAL \_start  
  
\_start:  
mov eax, msg\_func  
call sprintLF  
  
pop ecx  
pop edx  
sub ecx, 1  
mov esi, 0  
  
next:  
cmp ecx, 0  
jz \_end  
pop eax  
call atoi  
  
mov ebx, 6  
mul ebx  
add eax, 13  
add esi, eax  
  
loop next  
  
\_end:  
mov eax, msg\_result  
call sprint  
mov eax, esi  
call iprintLF  
call quit

Запускаем программу в режике отладичка и пошагово просматриваем изменение значений регистров через i r. При выполнении инструкции mul ecx можно заметить, что результат умножения записывается в регистр eax, но также меняет и edx. Значение регистра ebx не обновляется напрямую, поэтому результат программа неверно подсчитывает функцию (рис. 20):

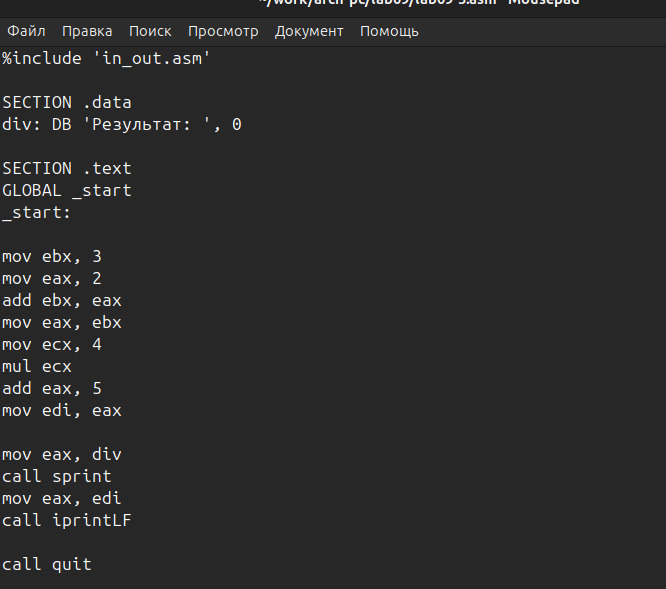


Рис. 20: снимок экрана

%include 'in\_out.asm'  
  
SECTION .data  
div: DB 'Результат: ', 0  
  
SECTION .text  
GLOBAL \_start  
\_start:  
  
mov ebx, 3  
mov eax, 2  
add ebx, eax  
mov eax, ebx  
mov ecx, 4  
mul ecx  
add eax, 5  
mov edi, eax  
  
mov eax, div  
call sprint  
mov eax, edi  
call iprintLF  
  
call quit

Исправляем найденную ошибку, теперь программа верно считает значение функции (рис. 21):

Рис. 21: снимок экрана

Рис. 21: снимок экрана

# 4 Выводы

При выполнении лабораторной работы были приобретены навыки написания программ с использованием подпрограмм. Ознакомились с методами отладки при помощи GDB и его основными возможностями.