Лабораторная работа №9

Понятие подпрограммы. Отладчик GDB.

Тютрюмова Анжелина

Содержание

# 1 Цель работы

Целью работы является приобретение навыков написания программ с использованием подпрограмм. Знакомство с методами отладки при помощи GDB и его основными возможностями.

# 2 Выполнение лабораторной работы

Создала и перешла в директорию для лабораторной работы создала файл lab9-1.asm (рис. 1).

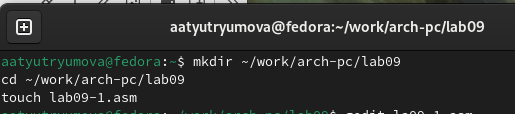


Рис. 1: Папка для лабораторной работы

Переписала код с лабараторной работы(рис. 2).

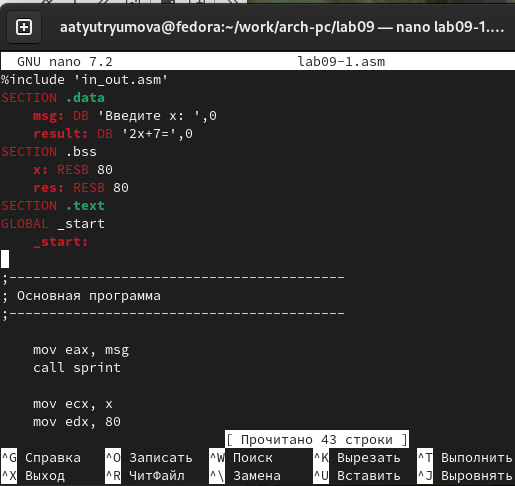


Рис. 2: Листинг кода

Листинг кода:

%include 'in\_out.asm'  
SECTION .data  
 msg: DB 'Введите x: ',0  
 result: DB '2x+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 "2x+7"  
  
\_calcul:  
 mov ebx,2  
 mul ebx  
 add eax,7  
 mov [res],eax  
ret ; выход из подпрограммы

Создала исполняемый файл и запустила его. (рис. 3).

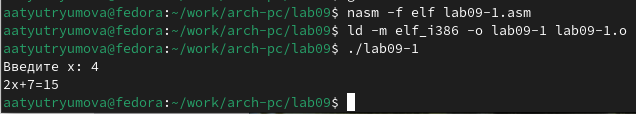


Рис. 3: Результат выполнения

Создала файл lab09-2.asm с кодом из лабараторной работы.(Программа печати сообщения Hello world!)

Листинг кода:

SECTION .data  
 msg1: db "Hello, ",0x0  
 msg1Len: equ $ - msg1  
 msg2: db "world!",0xa  
 msg2Len: equ $ - msg2  
SECTION .text  
 global \_start  
\_start:  
 mov eax, 4  
 mov ebx, 1  
 mov ecx, msg1  
 mov edx, msg1Len  
 int 0x80  
 mov eax, 4  
 mov ebx, 1  
 mov ecx, msg2  
 mov edx, msg2Len  
 int 0x80  
 mov eax, 1  
 mov ebx, 0  
 int 0x80

Получила исполняемый файл. Для работы с GDB добавила в исполняемый файл отладочную информацию, трансляцией с ключом ‘-g’. Загрузила исполняемый файл в отладчике gdb. (рис. 4).

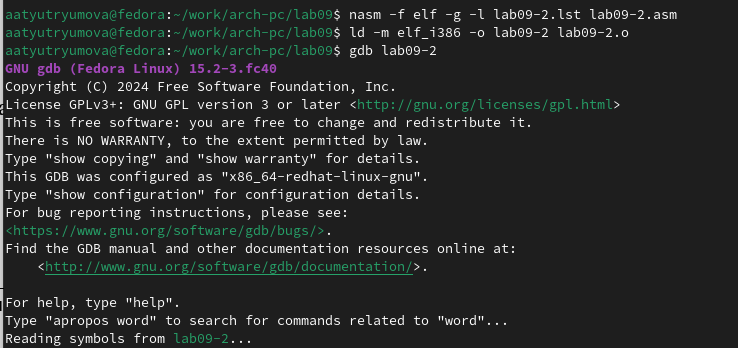


Рис. 4: Отладчике gdb

Проверила работу программы, запустив ее в оболочке GDB с помощью команды run. (рис. 5).

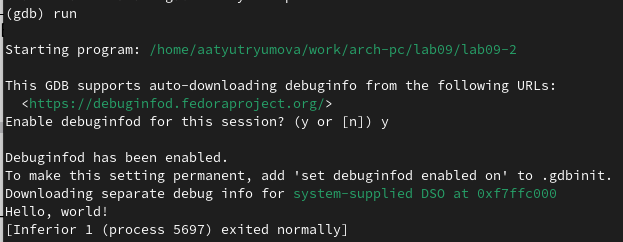


Рис. 5: Результат выполнения

Для более подробного анализа программы установила брейкпоинт на метку \_start, с которой начинается выполнение любой ассемблерной программы, и запустила её. ( рис. 6).

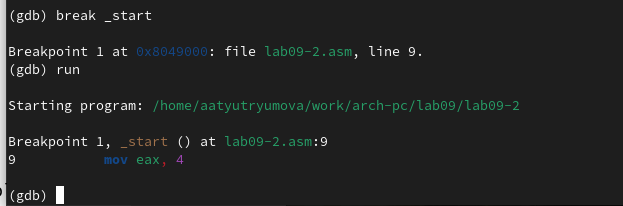


Рис. 6: Добавление брейкпоинта

Посмотрела дисассимилированный код программы с помощью команды disassemble начиная с метки \_start (рис. 7).



Рис. 7: Дисассимилированный код

Переключитесь на отображение команд с Intel’овским синтаксисом, введя команду set disassembly-flavor intel рис. 8).

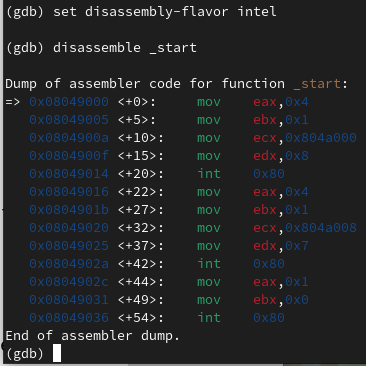


Рис. 8: Отображение команд с Intel’овским синтаксисом

Различие отображения синтаксиса машинных команд в режимах ATT и Intel состоит в том, что менется положение название регистраи его значения. Включила режим псевдографики для более удобного анализа программы (рис. 9).

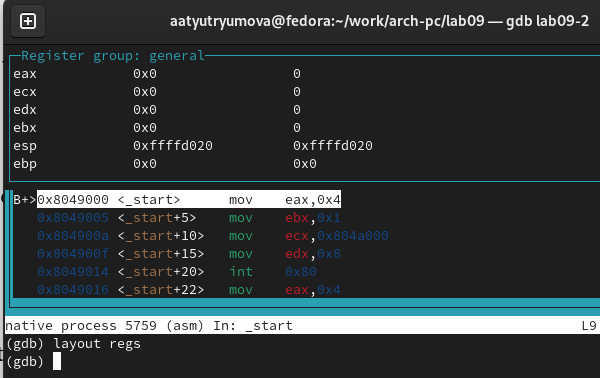


Рис. 9: Результат работы

Установила еще одну точку остановки по адресу инструкции. Для этого определила адрес предпоследней инструкции (mov ebx,0x0) и установила точку остановки. Посмотрела информацию о всех установленных брейкпоинтах. (рис. 10, рис. 11, рис. 12).

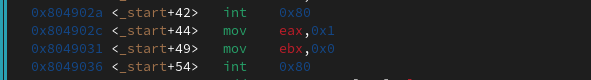


Рис. 10: Адрес предпоследней инструкции

Рис. 11: Установка брейкпоинта

Рис. 11: Установка брейкпоинта

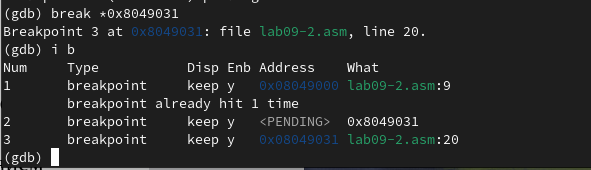


Рис. 12: Информация о брейпоинтах

Выполнила 5 инструкций с помощью команды si и проследите за изменением значений регистров. Значения регистров eax и eip (рис. 13).

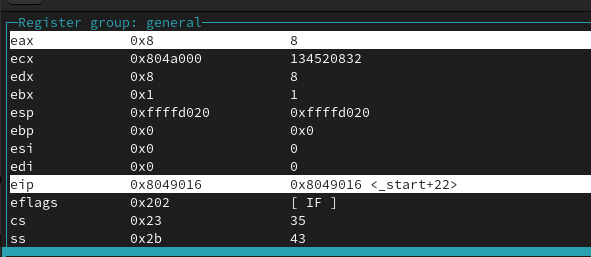


Рис. 13: Информация о брейпоинтах

Посмотрела значения переменных msg1 и msg2 по имени (рис. 14).

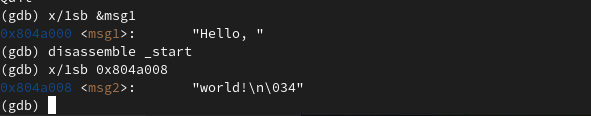


Рис. 14: Значение переменных msg1 и msg2

Изменила первый символ переменной msg1 (рис. 15).

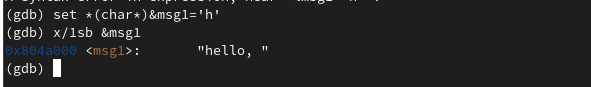


Рис. 15: Переменная msg1

Изменила первый символ переменной msg2 (рис. 16).

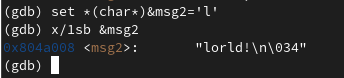


Рис. 16: Переменная msg2

С помощью команды set изменила значение регистра ebx (рис. 17).

Рис. 17: Значение регистра ebx

Рис. 17: Значение регистра ebx

Завершила выполнение программы с помощью команды continue и выйшла из GDB с помощью команды quit (рис. 18).

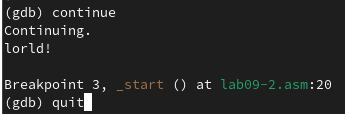


Рис. 18: Завершение работы

Скопировала файл lab8-2.asm, созданный при выполнении лабораторной работы №8, с программой выводящей на экран аргументы командной строки в файл с именем lab09-3.asm. Создала исполняемый файл и загрузила в gdb программу с аргументами и ключом –args. (рис. 19).

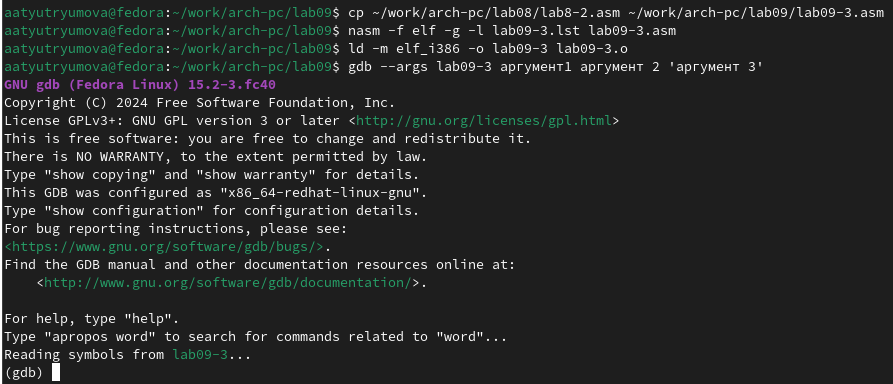


Рис. 19: lab09-3.asm

Установила точку останова перед первой инструкцией в программе и запустила ее (рис. 20).

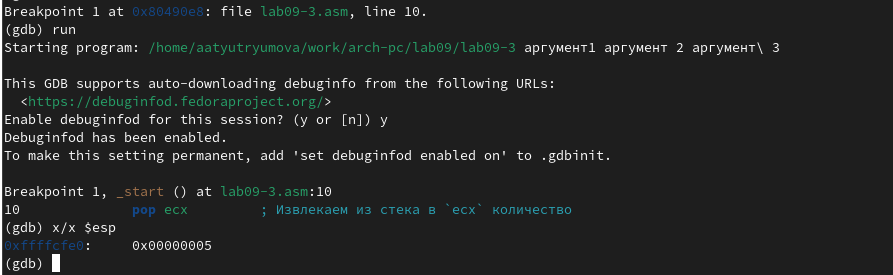


Рис. 20: Информация о брейпоинтах

Посмотрела позиции стека, в которых распологаются аргументы программы (рис. 21).

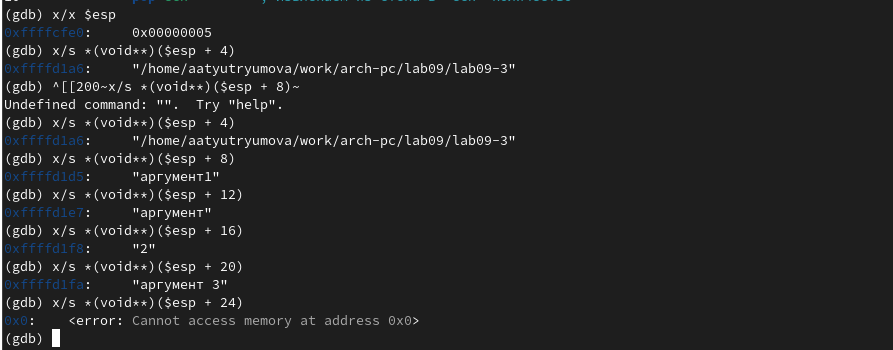


Рис. 21: Позиции стека и аргументы

# 3 Выполнение самостоятельной работы

Преобразовала программу из лабораторной работы №8 (Задание №1 для самостоятель- ной работы), реализовав вычисление значения функции f(x)=5(2+x) как подпрограмму. (рис. 22).

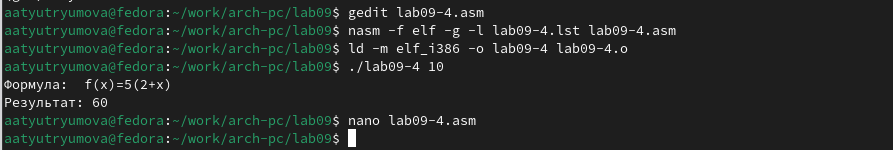


Рис. 22: Выполнение программы

Листинг кода:

%include 'in\_out.asm'  
  
SECTION .data  
 msg db "Результат: ",0  
 formula db "Формула: f(x)=5(2+x)",0  
  
SECTION .bss  
 res: RESB 80  
  
SECTION .text  
global \_start  
  
\_start:  
 pop ecx ; Извлекаем из стека в `ecx` количество  
 ; аргументов (первое значение в стеке)  
 pop edx ; Извлекаем из стека в `edx` имя программы  
 ; (второе значение в стеке)  
 sub ecx,1 ; Уменьшаем `ecx` на 1 (количество  
 ; аргументов без названия программы)  
next:  
 cmp ecx,0h ; проверяем, есть ли еще аргументы  
 jz \_end ; если аргументов нет выходим из цикла  
 ; (переход на метку `\_end`)  
 pop eax ; иначе извлекаем следующий аргумент из стека  
 call atoi ; преобразуем символ в число  
   
 call \_calcul  
 add [res],eax ; добавляем к промежуточной сумме  
 ; след. аргумент `esi=esi+eax`  
 loop next ; переход к обработке следующего аргумента  
   
\_end:  
 mov eax, formula; вывод сообщения "Формула: "  
 call sprintLF  
 mov eax, msg ; вывод сообщения "Результат: "  
 call sprint  
 mov eax, [res] ; записываем сумму в регистр `eax`  
 call iprintLF ; печать результата  
   
 call quit ; завершение программы  
   
;------------------------------------------  
; Подпрограмма вычисления  
; функции "f(x)=5(2+x)"  
  
\_calcul:  
 add eax, 2 ; Прибавляем 2  
 mov ebx, 5 ; ebx = 5  
 mul ebx ; Умножаем на 5  
 ret

Проверила, что программа при запуске дает неверный результат (рис. 23)

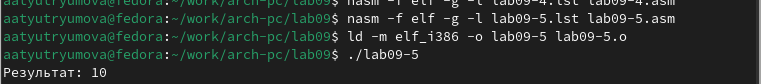


Рис. 23: Выполнение программы

С помощью отладчика GDB, анализируя изменения значений регистров, определила ошибку и исправила ее (рис. 24)

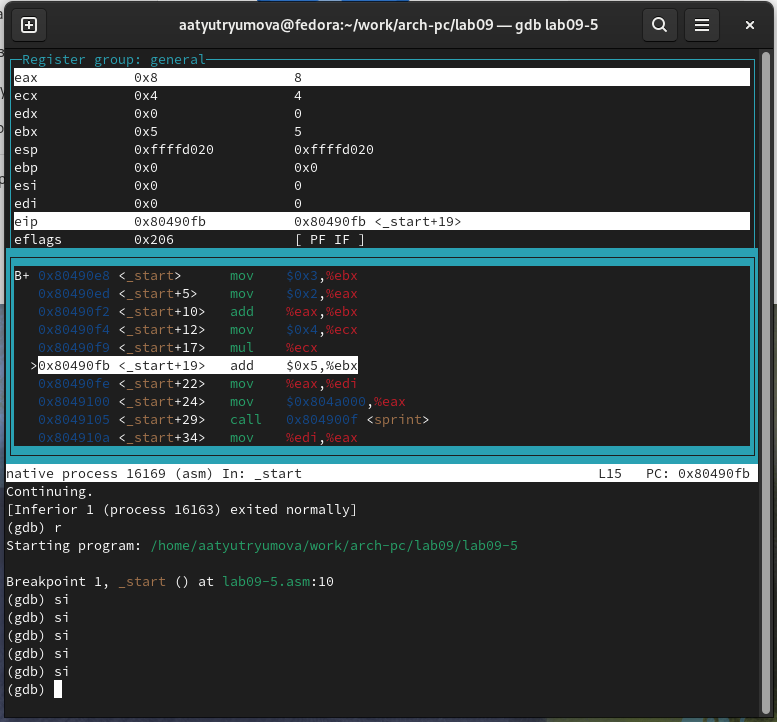


Рис. 24: Отладчика GDB

Листинг кода(исправленный):

%include 'in\_out.asm'  
 SECTION .data  
 div: DB 'Результат: ',0  
   
 SECTION .text  
 GLOBAL \_start  
 \_start:  
   
 ; ---- Вычисление выражения (3+2)\*4+5  
 mov ebx,3  
 mov eax,2  
 add eax,ebx  
 mov ecx,4  
 mul ecx  
 add eax,5  
 mov edi,eax  
   
 ; ---- Вывод результата на экран  
 mov eax,div  
 call sprint  
 mov eax,edi  
 call iprintLF  
 call quit

Проверила корректность исполнения программы (рис. 25)

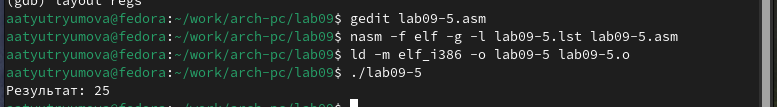


Рис. 25: Выполнение программы

# 4 Выводы

Выполнив данную лабараторную работу, я обрела навыки написания программ с использованием подпрограмм. И ознакомилась с методами отладки при помощи GDB и его основными возможностями.