Oтчёта по лабораторной работе 10

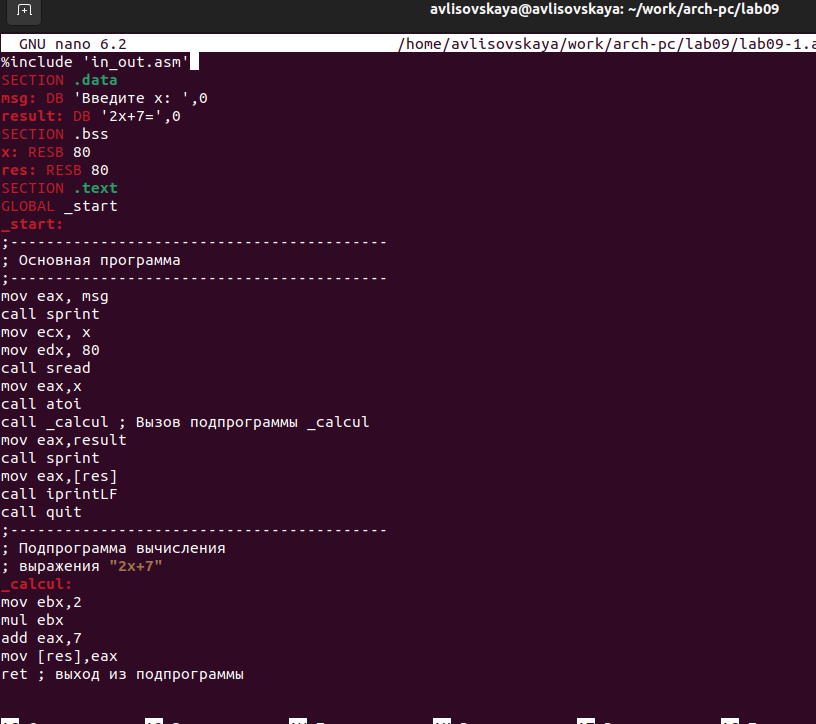
Лисовская Арина Валерьевна

Содержание

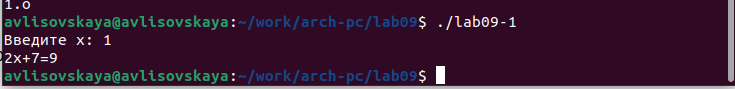
# 1 Цель работы

Освоить работу с подпрограммами и отладчиком gdb.

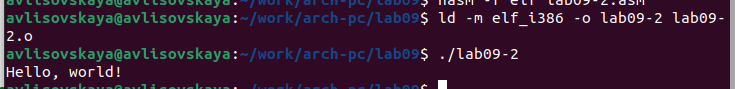
# 2 Выполнение лабораторной работа

Создадим рабочую директорию и файл. Запишем туда программу из листинга, исправив опечатки. (рис. [??]). 

напишем программу, имитирующую сложную функцию и проверяем ее работу (рис. [[1](#fig:002)])



Создадим файл lab09-2.asm и посмотрим, как она работает. Так же проассемблируем его с другими ключами, чтобы была возможность открыть этот файл через gdb. (рис. [??])



-

Откроем lab09-2 с помощью gdb. Запустим ее там(рис. [??])



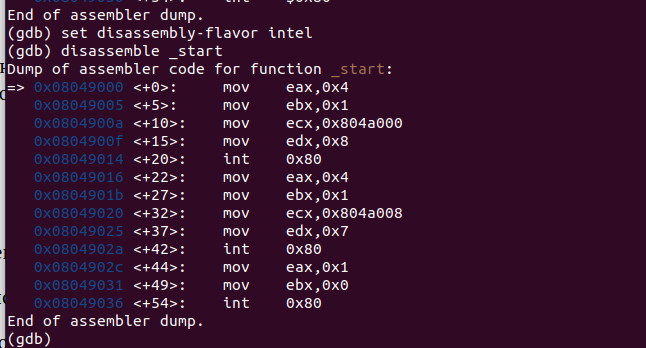
-

Поставим точку останова(breakpoint) на метке \_start. Посмотрим дизассемеблированный код, начиная с этой метки. (рис. [??])



-

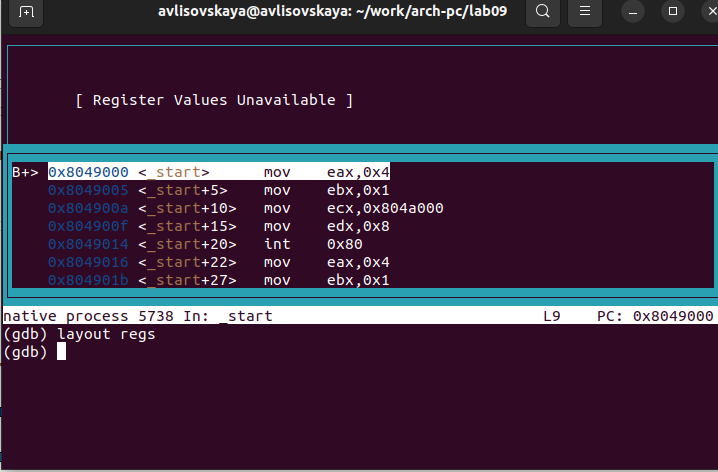
Так же посмотрим как выглядит дизассемблированный код c синтаксисом Intel (рис. [??])



-

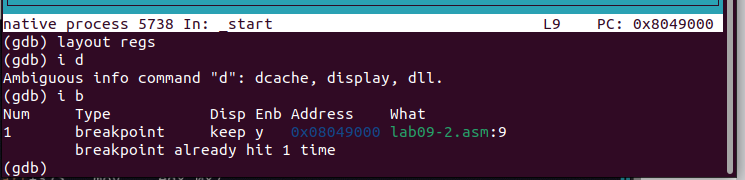
В представлении ATT в виде 16-ричного числа записаны первые аргументы всех комманд, а в представлении intel так записываются адреса вторых аргумантов.

включим режим псевдографики, с помощью которго отбражается код программы и содержимое регистров(рис. [??])

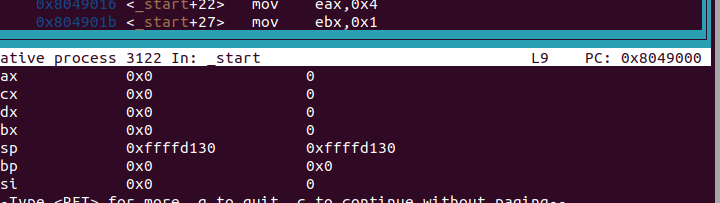


-

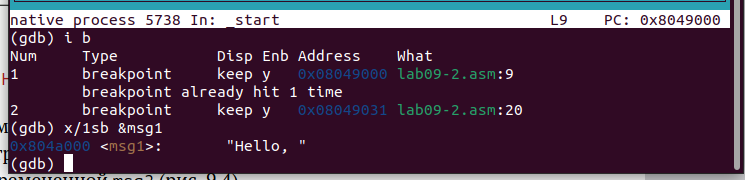
Посмотрим информацию о наших точках останова. Сделать это можно коротко командой i b (рис. [??])



-

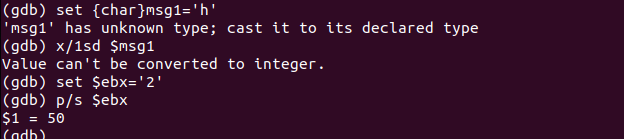
Так же можно выводить значения регистров. Делается это командой i r. Псевдографика предствалена на (рис. [??]) 

В отладчике можно вывести текущее значение переменных. Сделать это можно например по имени (рис. [??]) или по адресу (рис. [??])



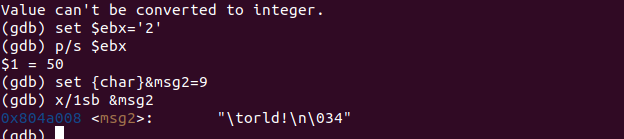
-

Так же отладчик позволяет менять значения переменных прямо во время выполнения программы (рис. [??])



-

Здесь тоже можно обращаться по адресам переменных(рис. [??]). здесь был заменен первый символ переменной msg2 на символ отступа.

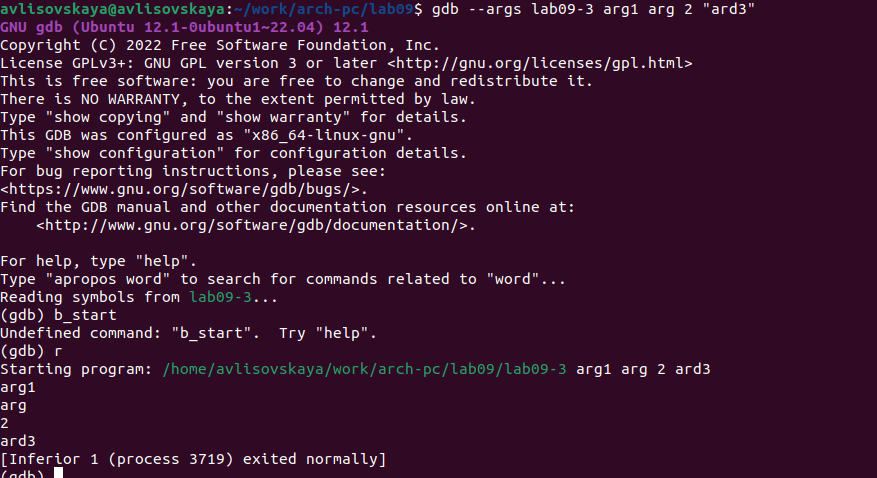


-

Однако при попытке задать строчное значение, происходит ошибка.

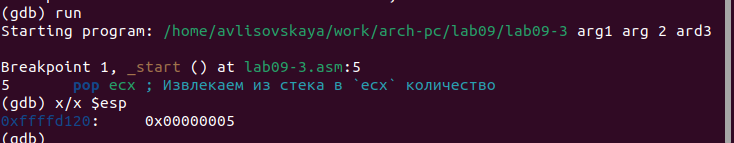
Завершим работу в gdb командами continue, она закончит выполнение программы, и exit, она завершит сеанс gdb.

Скопируем файл из лабораторной 9, переименуем и создадим исполняемый файл. Откроем отладчик и зададим аргументы. Создадим точку останова на метке \_start и запустим программу(рис. [??])



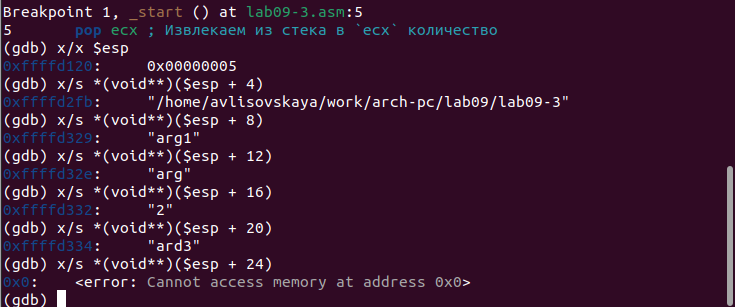
-

Посмотрим на содержимое того,что расположено по адрессу, находящемуся в регистре esp (рис. [??])



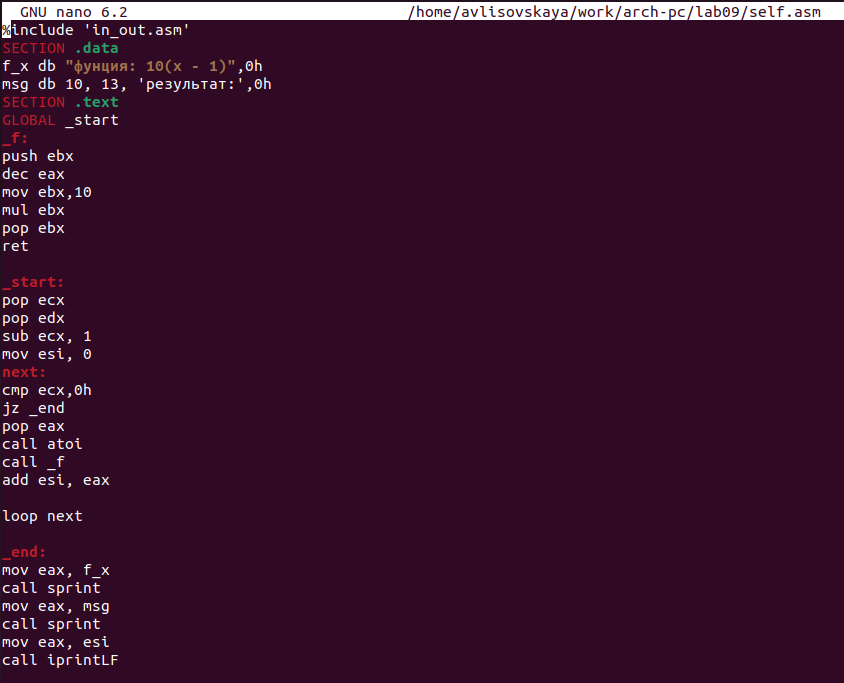
-

Далее посмотрим на все остальные аргументы в стеке. Их адреса распологаются в 4 байтах друг от друга(именно столько заниемает элемент стека) (рис. [??])



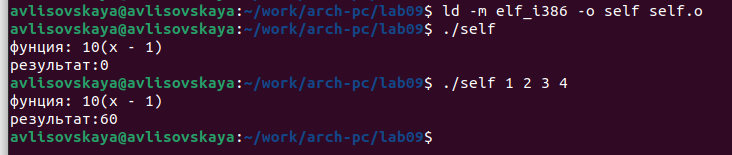
-

#Задания для самостоятельной работы Программа из лабороторной 9, но с использованием подпрограмм (рис. [??])



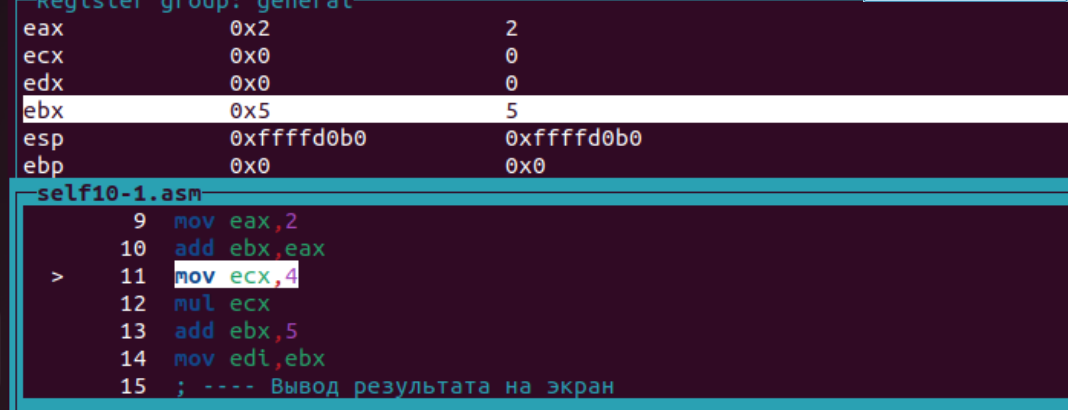
-

и проверка ее работоспособности(рис. [??])

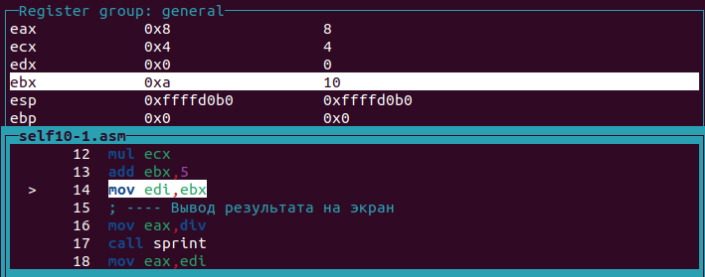


-

Просмотр регистров, для поиска ошибки в программе из листинга 10.3 (рис. [??]) и (рис. [??])



-

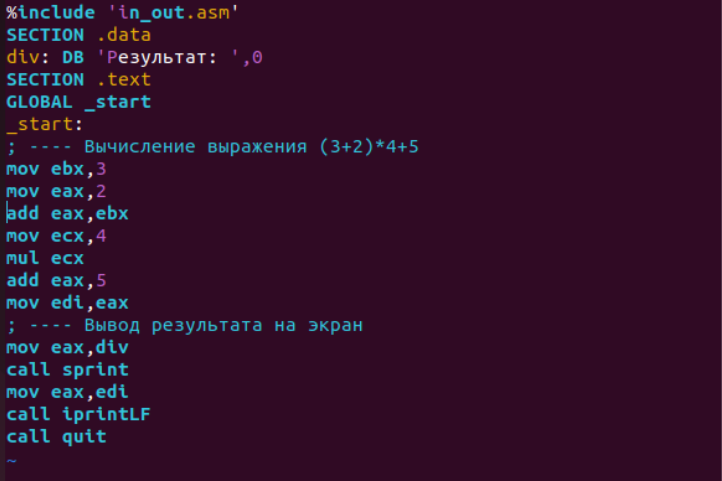


-

Ошибка была в сторках

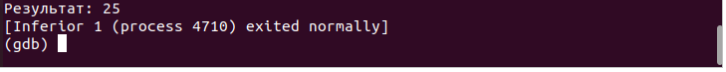
add ebx,eax mov ecx,4 mul ecx add ebx,5 mov edi,ebx

правильно работающая программа представлена на (рис. [??])



-

Проверка корректронсти работы программы, после исправлений (рис. [??])



-

# 3 Выводы

В результате выполнения работы, я научился организовывать код в подпрограммы и познакомился с базовыми функциями отладчика gdb.