Отчёт по лабораторной работе №9

Дисциплина: архитектура компьютера

Незами Ахмад Белал

Содержание

# 1 Цель работы

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

# 2 Выполнение лабораторной работы

**1**

Создадим рабочую директорию и файл.(рис. [[1](#fig:001)])

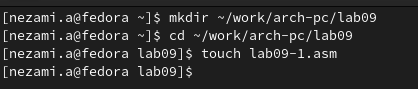


Figure 1: Создаем директорию и файл

**2**

Напишем программу, имитирующую сложную функцию. Функции назовем \_calul и subcalcul.(рис. [[2](#fig:002)])

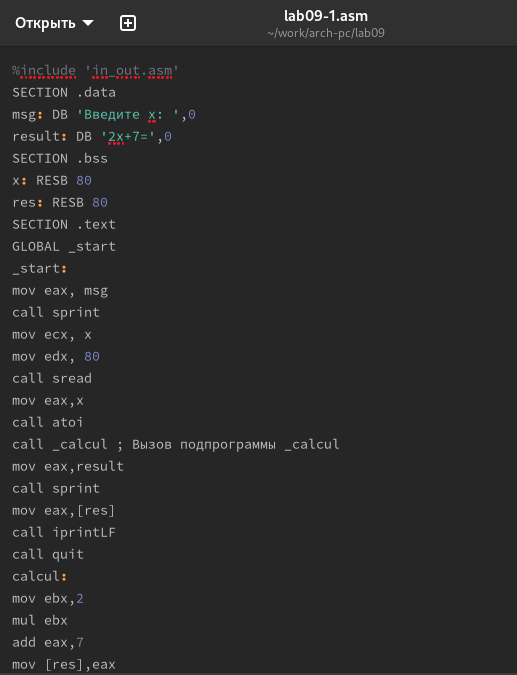


Figure 2: Запись программы

**3**

Проверим ее работу (рис. [-[3](#fig:003))

Figure 3: Проверка

Figure 3: Проверка

**4**

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. [[4](#fig:004)])

Figure 4: Создание файла lab09-2.asm

Figure 4: Создание файла lab09-2.asm

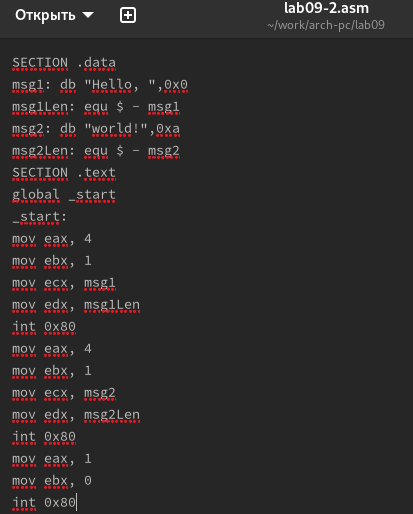


Figure 5: Запись программы

**5**

Откроем lab09-2 с помощью gdb. Запустим ее там(рис. [[6](#fig:006)])

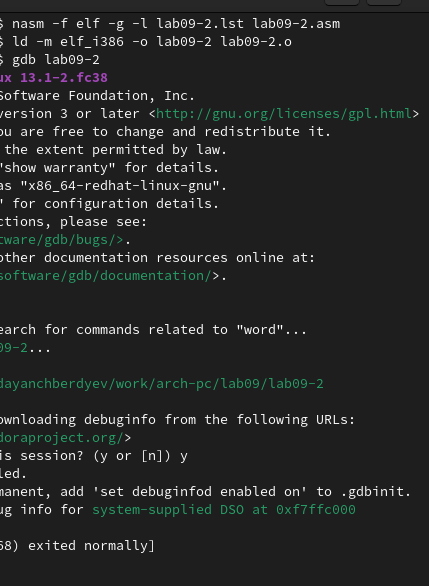


Figure 6: Запуск программы

**6**

Поставим точку останова(breakpoint) на метке \_start. Посмотрим дизассемеблированный код, начиная с этой метки. (рис. [[7](#fig:007)])

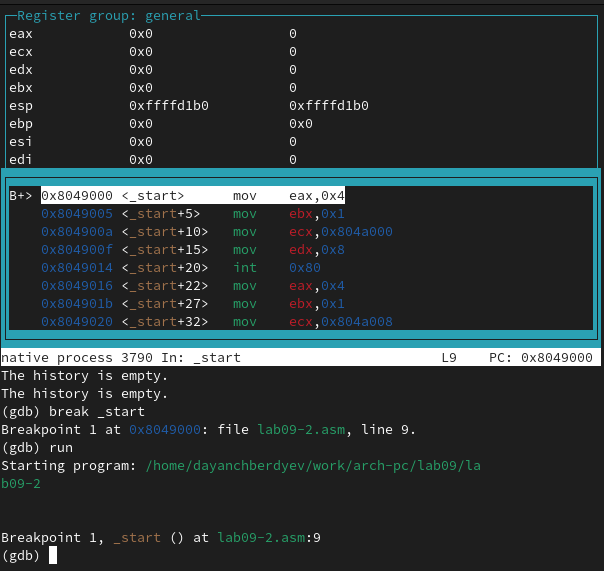


Figure 7: Поставим точку останова(breakpoint) на метке \_start

**7**

Так же посмотрим как выглядит дизассемблированный код c синтаксисом Intel (рис. [[9](#fig:008)])

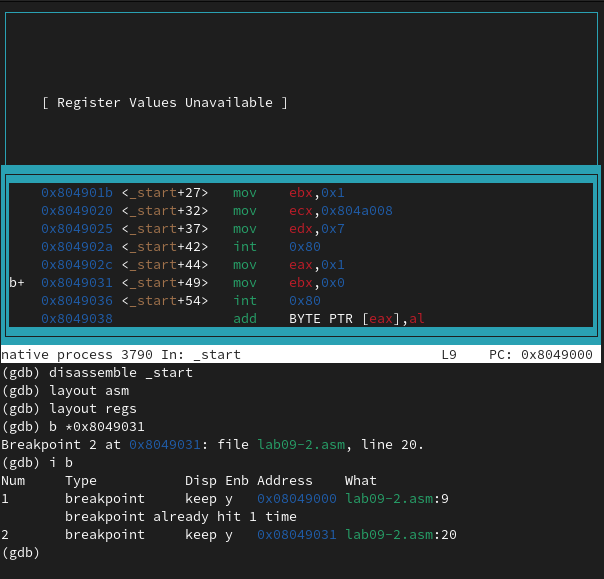


Figure 8: дизассемблированный код

**8**

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.

Посмотрим информацию о наших точках останова. Сделать это можно коротко командой i b (рис. [[9](#fig:008)])

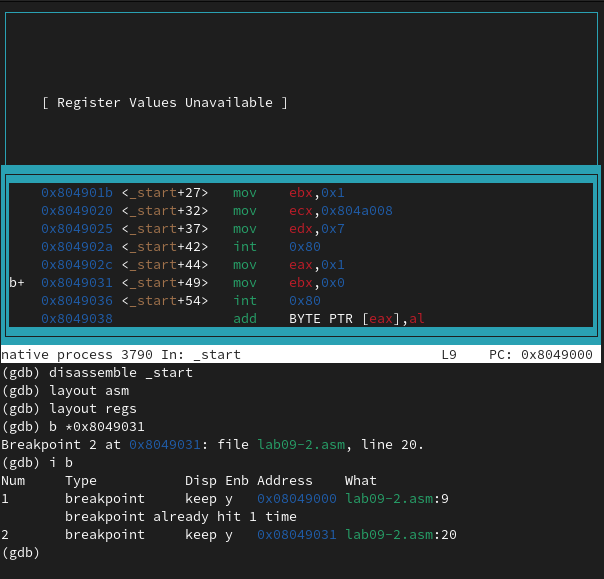


Figure 9: Команда i b

**9**

В отладчике можно вывести текущее значение переменных. Сделать это можно например по имени или по адресу (рис. [[10](#fig:009)])

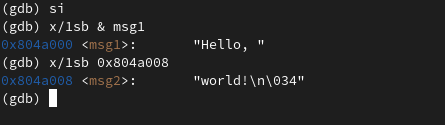


Figure 10:

**10**

Так же отладчик позволяет менять значения переменных прямо во время выполнения программы (рис. [[11](#fig:010)])

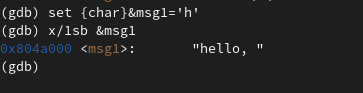


Figure 11:

**11**

Здесь тоже можно обращаться по адресам переменных(рис. [[12](#fig:011)]). здесь был заменен первый символ переменной msg2 на символ отступа.

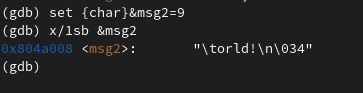


Figure 12:

**12**

Выоводить можно так же содержимое регисторов. Выведем значение edx в разных форматах: строчном, 16-ричном, двоичном(рис. [[14](#fig:012)])

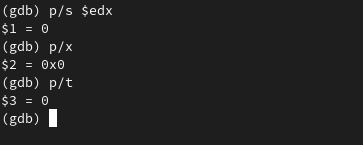


Figure 13:

**13**

Как и переменным, регистрам можно задавать значения.(рис. [??])

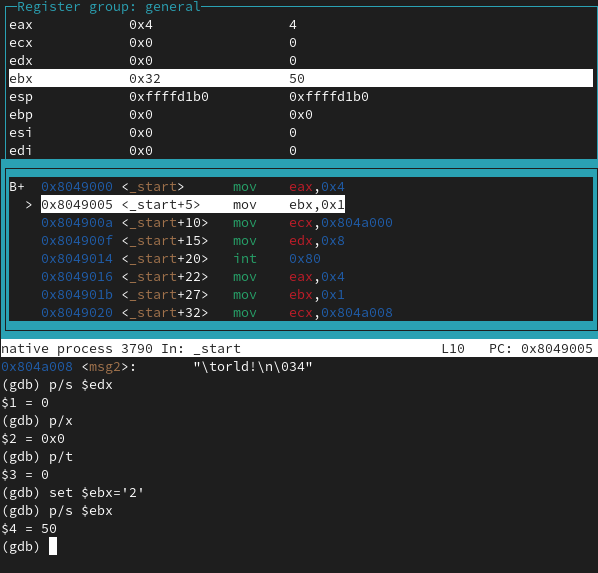


Figure 14:

**14**

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. [[15](#fig:014)])

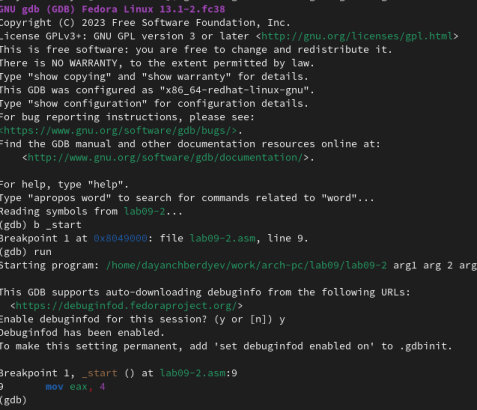


Figure 15:

**15**

Посмотрим на содержимое того,что расположено по адрессу, находящемуся в регистре esp (рис. [[16](#fig:015)])

Figure 16: 

Figure 16:

**16**

Далее посмотрим на все остальные аргументы в стеке. Их адреса распологаются в 4 байтах друг от друга(именно столько заниемает элемент стека) (рис. [[17](#fig:016)])



Figure 17:

# 3 Задания для самостоятельной работы

**17**

Программа из лабороторной 9, но с использованием подпрограмм (рис. [[18](#fig:017)])

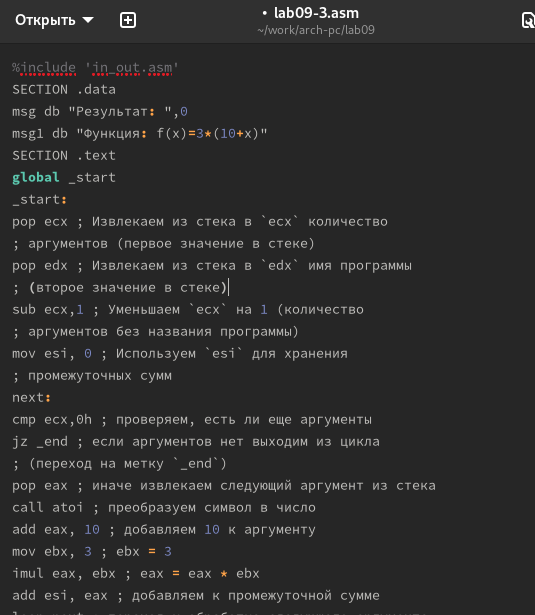


Figure 18:

**18**

Проверка ее работоспособности(рис. [[19](#fig:018)])

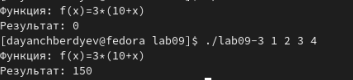


Figure 19:

Просмотр регистров, для поиска ошибки в программе из листинга

Ошибка была в сторках

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

Правильно работающая программа представлена на (рис. [[20](#fig:019)])

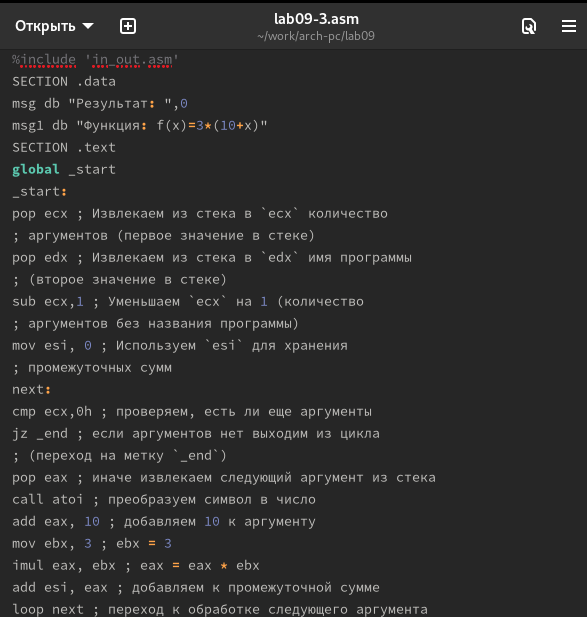


Figure 20: Программа работает правильно

Проверка корректронсти работы программы, после исправлений (рис. [[21](#fig:020)])

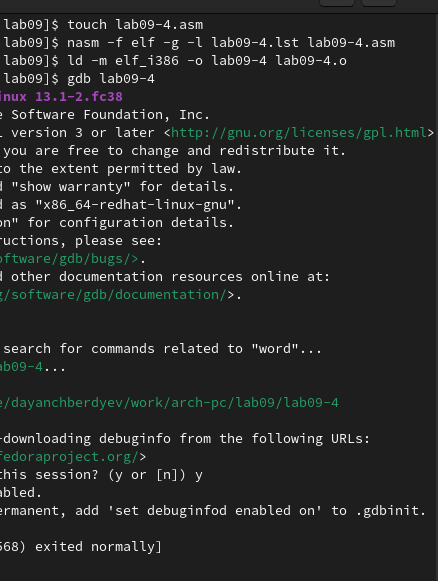


Figure 21: Проверка

# 4 Выводы

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