Отчёт по лабораторной работе №9

Простейший вариант

Метвалли Ахмед Фарг Набеех

Содержание

1 Цель работы 5

2 Выполнение лабораторной работы 6

3 Задания для самостоятельной работы 13

4 Выводы 15

Список иллюстраций

2.1

2.2 f(g(x))

2.3

2.4

2.5

2.6

2.7

2.8

2.9

2.10

2.11

2.12

2.13

2.14

2.15

2.16

2.17

2.18

2.19

3.1

3.2

3.3

3.4

3.5

3.6

Список таблиц

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

Создадим рабочую директорию и файл. Запишем туда программу из листинга,

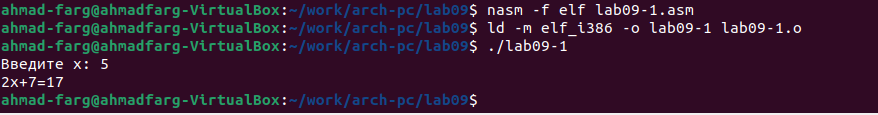
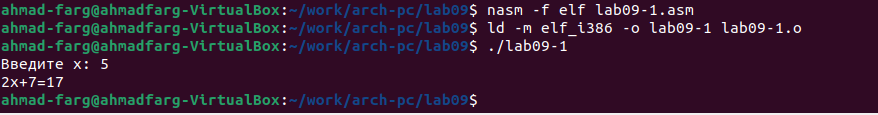


Рис. 2.1: напишем программу, имитирующую сложную функцию. Функции назовем \_calul и subcalcul(рис. 2.2)



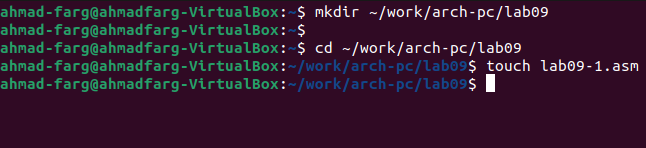


Рис. 2.2: f(g(x))

Проверим ее работу (рис. [-fig. 2.3)



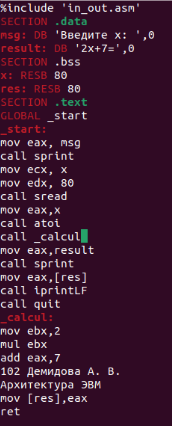


Рис. 2.3: -

Создадим файл lab10-2.asm и посмотрим, как она работает. Так же про ассемблируем его с другими ключами, чтобы была возможность открыть этот файл через gdb. (рис. 2.4)

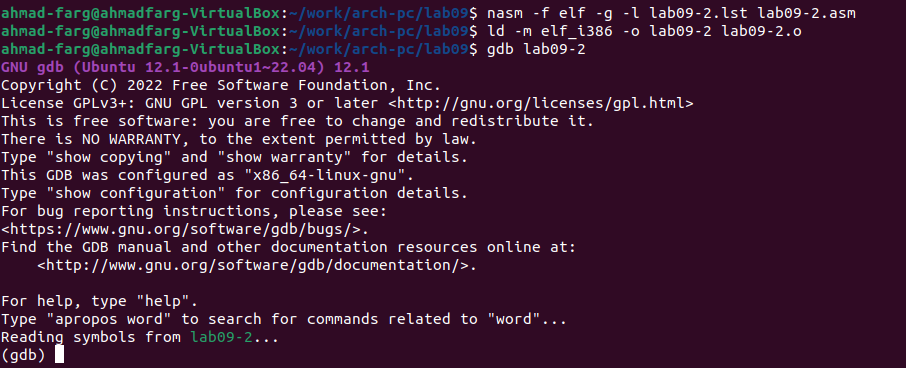
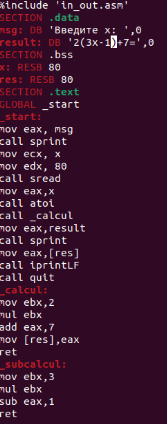


Рис. 2.4: -

Откроем lab10-2 с помощью gdb. Запустим ее там (рис. 2.5)



Рис. 2.5: -

Поставим точку останова(breakpoint) на метке \_start. Посмотрим дизассемблированный код, начиная с этой метки. (рис. 2.6)



Рис. 2.6: -

Так же посмотрим, как выглядит дизассемблированный код c синтаксисом Intel (рис. 2.7)



Рис. 2.7: -

В представлении ATT в виде 16-ричного числа записаны первые аргументы всех команд, а в представлении intel так записываются адреса вторых аргументов. Включим режим псевдографики, с помощью которого отображается код программы и содержимое регистров(рис. 2.8)

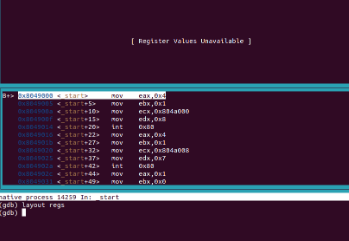


Рис. 2.8: -

Посмотрим информацию о наших точках останова. Сделать это можно коротко командой i b (рис. 2.9)



Рис. 2.9: -

добавим еще одну точку останова, но сделаем это по адресу (рис. 2.10)

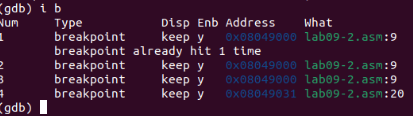


Рис. 2.10: -

Так же можно выводить значения регистров. Делается это командой i r. Псевдографика представлена на (рис. 2.11)

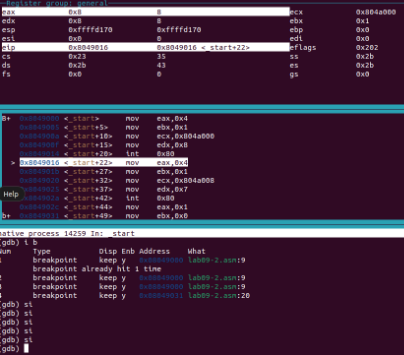


Рис. 2.11: -

В отладчике можно вывести текущее значение переменных. Сделать это можно, например по имени (рис. 2.12) или по адресу (рис. 2.13)



Рис. 2.12: -



Рис. 2.13: -

Так же отладчик позволяет менять значения переменных прямо во время выполнения программы (рис. 2.14)

Рис. 2.14: -

Здесь тоже можно обращаться по адресам переменных (рис. 2.15). здесь был заменен первый символ переменной msg2 на символ отступа.

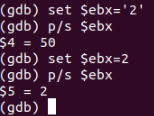


Рис. 2.15: -

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. 2.16)

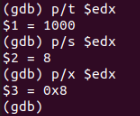


Рис. 2.16: -

Посмотрим на содержимое того, что расположено по адресу, находящемуся в регистре esp (рис. 2.17)





Рис. 2.17: -

Далеепосмотримнавсеостальныеаргументывстеке.Ихадресараспологаются в 4 байтах друг от друга (именно столько занимает элемент стека) (рис. 2.19)

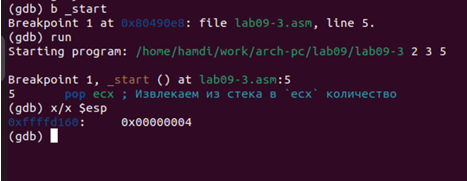
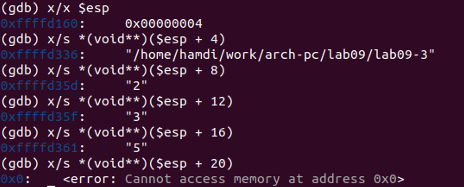


Рис. 2.19: -



3. Задания для самостоятельной работы

Программа из лабораторной 9, но с использованием подпрограмм (рис. 3.1)

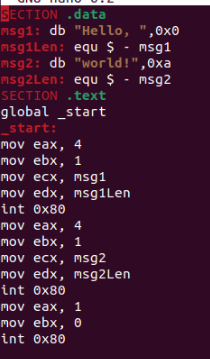


Рис. 3.1: -

и проверка ее работоспособности (рис. 3.2)



Рис. 3.2: -

Просмотр регистров, для поиска ошибки в программе из листинга 10.3 (рис. 3.3) и (рис. 3.4)

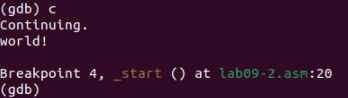
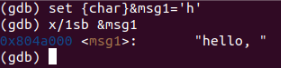


Рис. 3.3: -





Рис. 3.4: -



Ошибка была в сторках

add ebx,eax

mov ecx,4

mul ecx

add ebx,5

mov edi,ebx

правильно работающая программа представлена на (рис. 3.5)

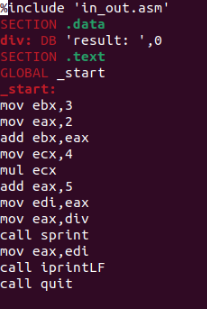


Рис. 3.5: - Проверка корректности работы

программы, после исправлений (рис. 3.6)



Рис. 3.6: -

4.Выводы

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