Отчёт по лабораторной работе 9

Дисциплина: Архитектура компьютера

Горобцова Арина Романовна НММбд-01-24

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

1. Создаем каталог для выполнения лабораторной работы № 9, переходим в него и создаем файл lab9-1.asm.
2. В качестве примера рассмотрим программу вычисления арифметического выражения с помощью подпрограммы calcul. В данном примере вводится с клавиатуры, а само выражение вычисляется в подпрограмме. (рис. 1) , (рис. 2)

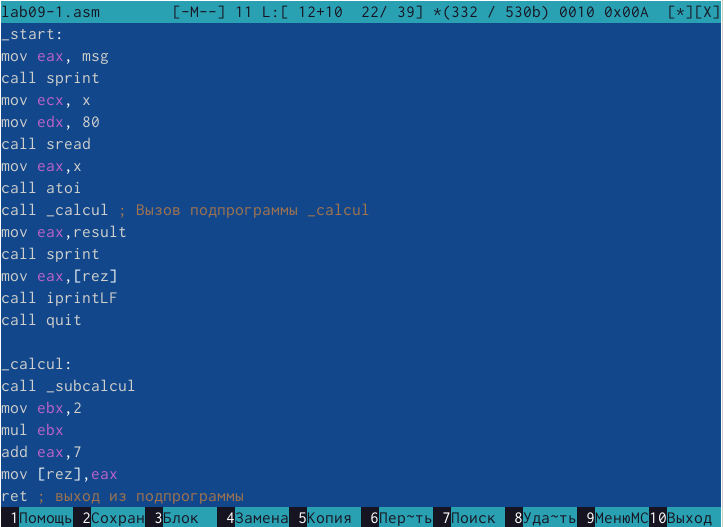


Рис. 1: Программа lab9-1.asm

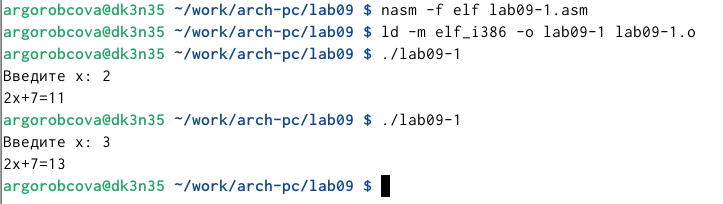


Рис. 2: Запуск программы lab9-1.asm

1. Изменяем текст программы, добавляем подпрограмму subcalcul в подпрограмму calcul, для вычисления выражения , где вводится с клавиатуры, . (рис. 3), (рис. 4)

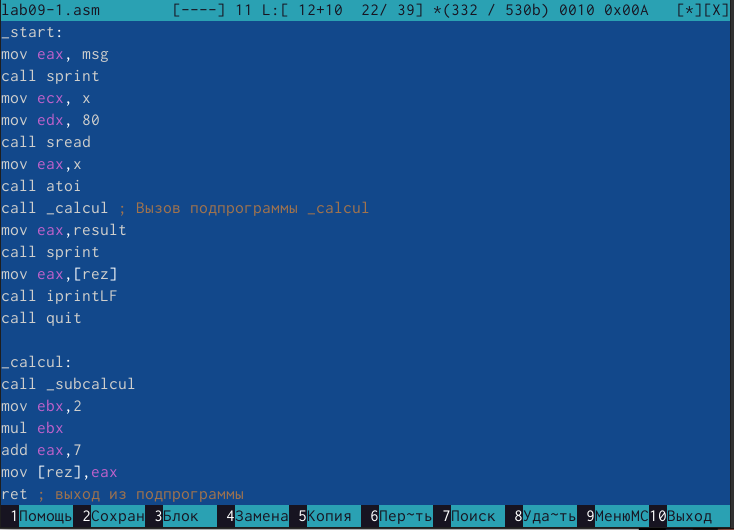


Рис. 3: Программа lab9-1.asm

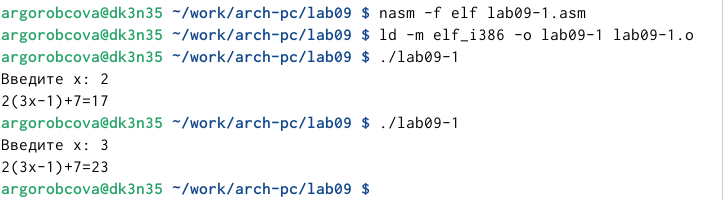


Рис. 4: Запуск программы lab9-1.asm

1. Создаем файл lab9-2.asm с текстом программы из Листинга 9.2. (Программа печати сообщения Hello world!). (рис. 5)

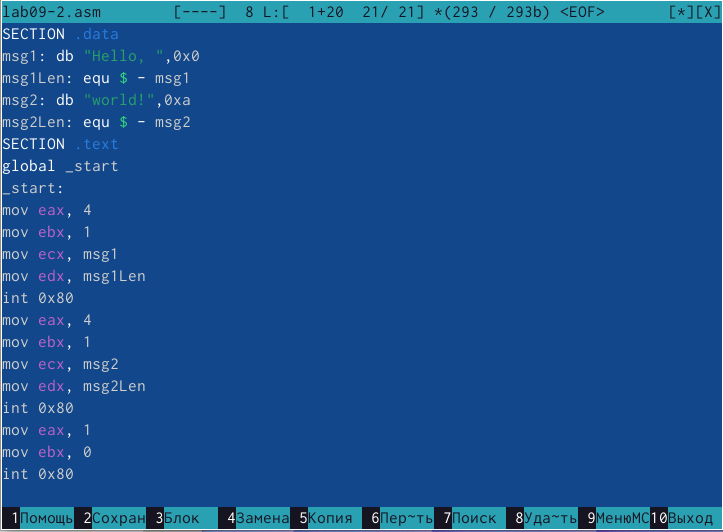


Рис. 5: Программа lab9-2.asm

Получаем исполняемый файл. Для работы с GDB в исполняемый файл необходимо добавить отладочную информацию, для этого трансляцию программ необходимо проводить с ключом ‘-g’.

Загружаем исполняемый файл в отладчик gdb. Проверяем работу программы, запустив ее в оболочке GDB с помощью команды run (сокращённо r). (рис. 6)

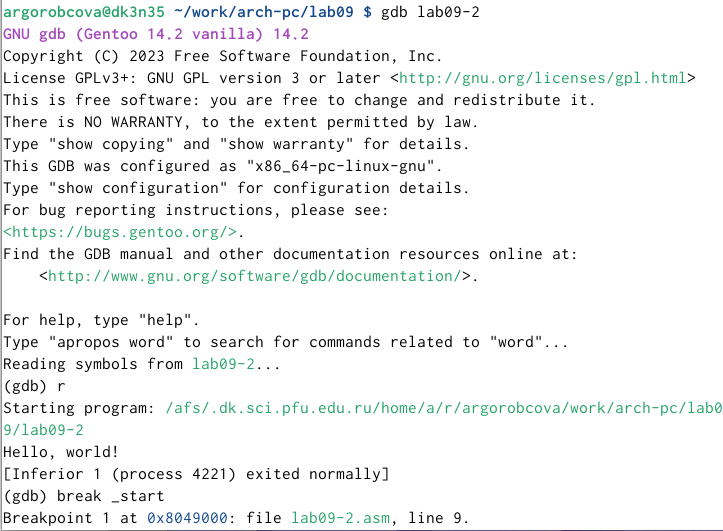


Рис. 6: Запуск программы lab9-2.asm в отладчике

Для более подробного анализа программы установите брейкпоинт на метку start, с которой начинается выполнение любой ассемблерной программы, и запускаем её. Посмотрим дизассемблированный код программы. (рис. 7), (рис. 8)

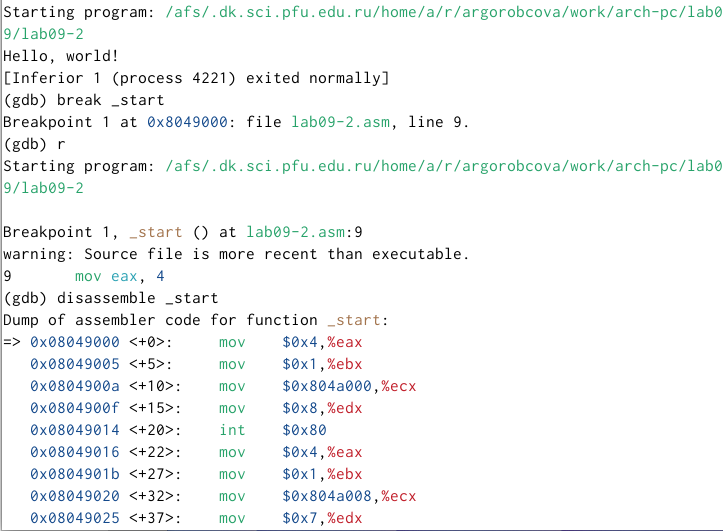


Рис. 7: Дизассемблированный код

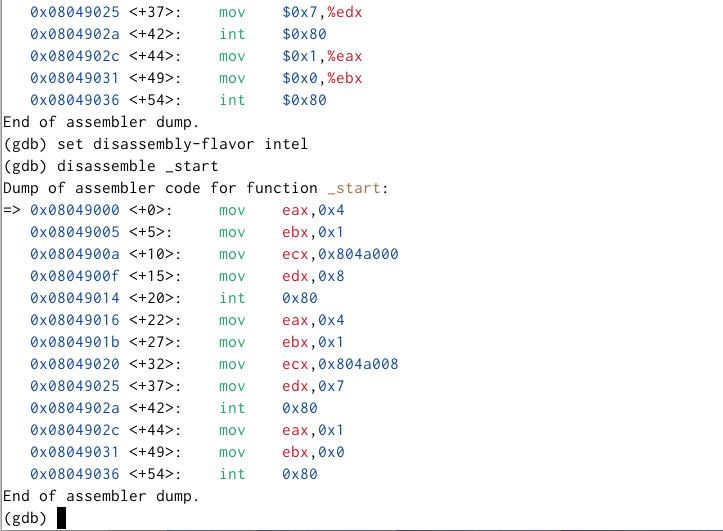


Рис. 8: Дизассемблированный код в режиме интел

На предыдущих шагах была установлена точка остановки по имени метки (\_start). Проверяем это с помощью команды info breakpoints (кратко i b). Установливаем еще одну точку остановки по адресу инструкции. Адрес инструкции можно увидеть в средней части экрана в левом столбце соответствующей инструкции. Определяем адрес предпоследней инструкции (mov ebx,0x0) и установливаем точку. (рис. 9)

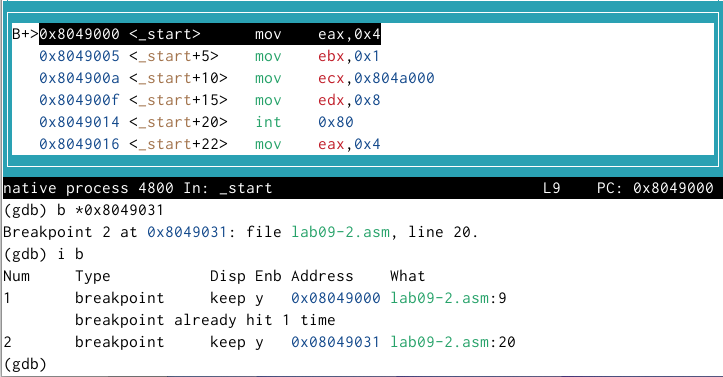


Рис. 9: Точка остановки

Отладчик может показывать содержимое ячеек памяти и регистров, а при необходимости позволяет вручную изменять значения регистров и переменных. Выполняем 5 инструкций с помощью команды stepi (или si) и смотрим за изменением значений регистров. (рис. 10), (рис. 11)

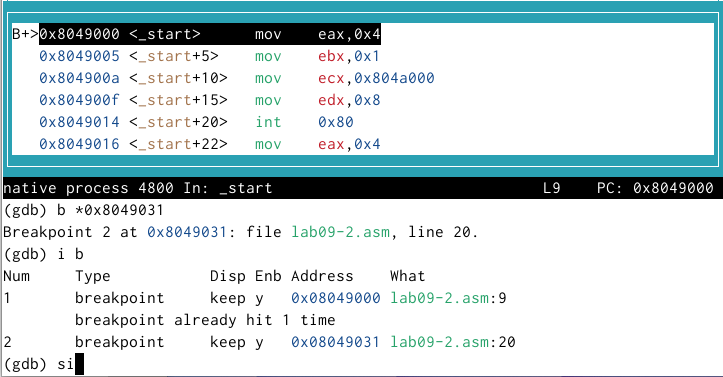


Рис. 10: Изменение регистров

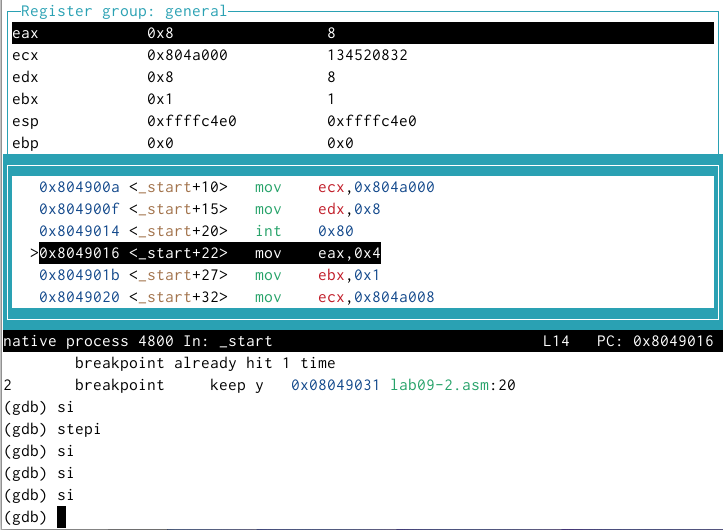


Рис. 11: Изменение регистров

Посмотрели значение переменной msg1 по имени. Посмотрели значение переменной msg2 по адресу.

Изменили значение для регистра или ячейки памяти можно с помощью команды set, задав ей в качестве аргумента имя регистра или адрес. Изменили первый символ переменной msg1.(рис. 12)

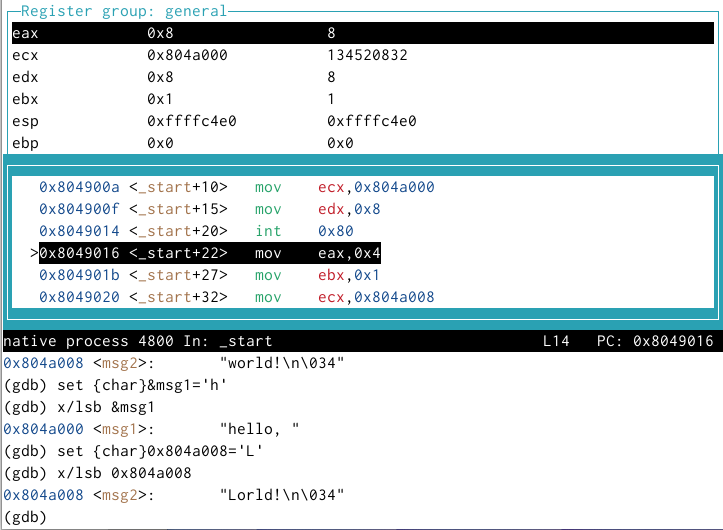


Рис. 12: Изменение значения переменной

Вывели в различных форматах (в шестнадцатеричном формате, в двоичном формате и в символьном виде) значение регистра edx. (рис. 13)

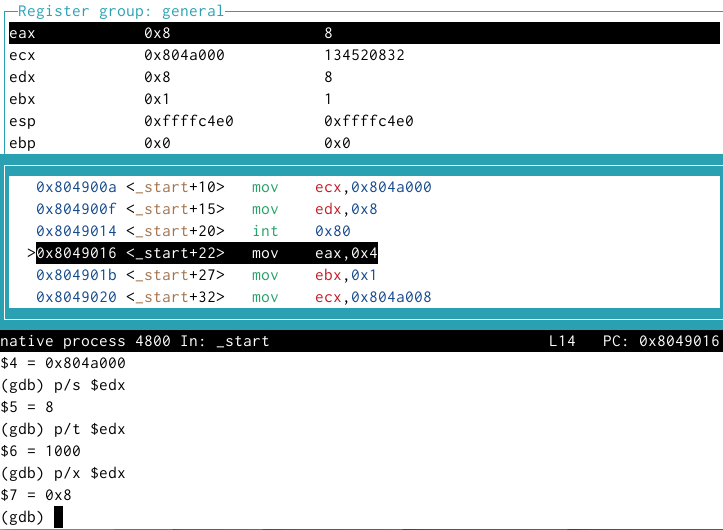


Рис. 13: Вывод значения регистра

С помощью команды set изменили значение регистра ebx (рис. 14)

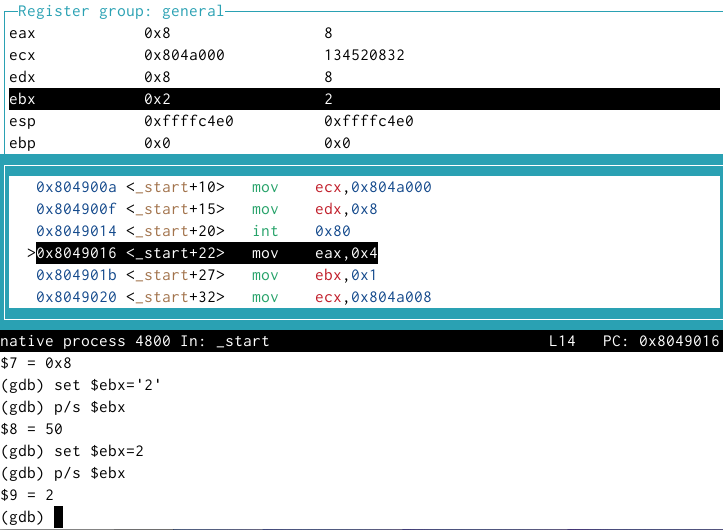


Рис. 14: Вывод значения регистра

1. Скопировали файл lab8-2.asm, созданный при выполнении лабораторной работы №8, с программой выводящей на экран аргументы командной строки. Создали исполняемый файл. Для загрузки в gdb программы с аргументами необходимо использовать ключ –args. Загрузили исполняемый файл в отладчик, указав аргументы.

Для начала установили точку останова перед первой инструкцией в программе и запустили ее.

Адрес вершины стека храниться в регистре esp и по этому адресу располагается число равное количеству аргументов командной строки (включая имя программы). Как видно, число аргументов равно 5 – это имя программы lab9-3 и непосредственно аргументы: аргумент1, аргумент, 2 и ‘аргумент 3’.

Посмотрели остальные позиции стека – по адесу [esp+4] располагается адрес в памяти где находиться имя программы, по адесу [esp+8] храниться адрес первого аргумента, по аресу [esp+12] – второго и т.д. (рис. 15)

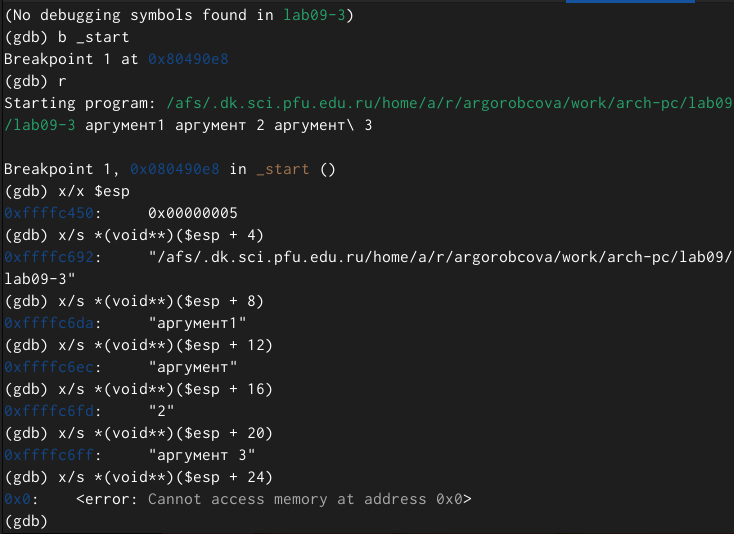


Рис. 15: Вывод значения регистра

Объясняем, почему шаг изменения адреса равен 4 ([esp+4], [esp+8], [esp+12] - потому что шаг равен размеру переменной - 4 байтам.

1. Преобразовали программу из лабораторной работы №8 (Задание №1 для самостоятельной работы), реализовав вычисление значения функции f(x) как подпрограмму. (рис. 16), (рис. 17)

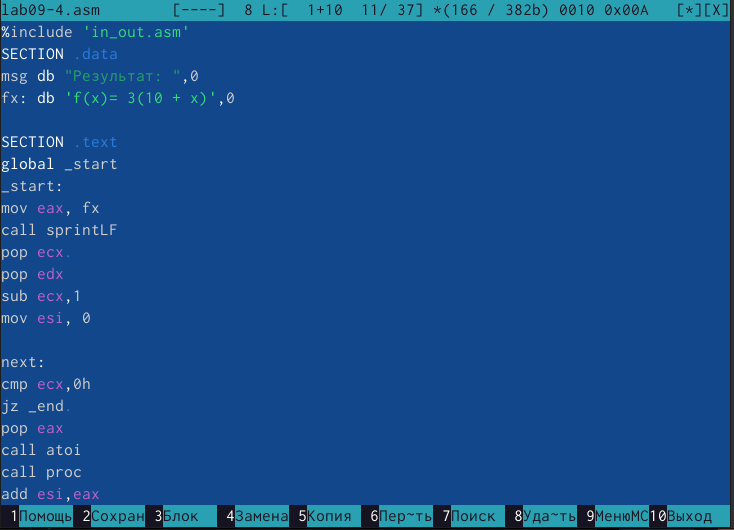


Рис. 16: Программа lab9-4.asm

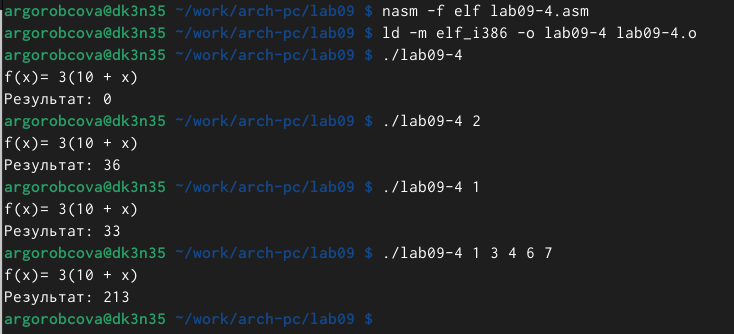


Рис. 17: Запуск программы lab9-4.asm

1. В листинге приведена программа вычисления выражения . При запуске данная программа дает неверный результат. Проверили это. С помощью отладчика GDB, анализируя изменения значений регистров, определяем ошибку и исправляем ее. (рис. 18)

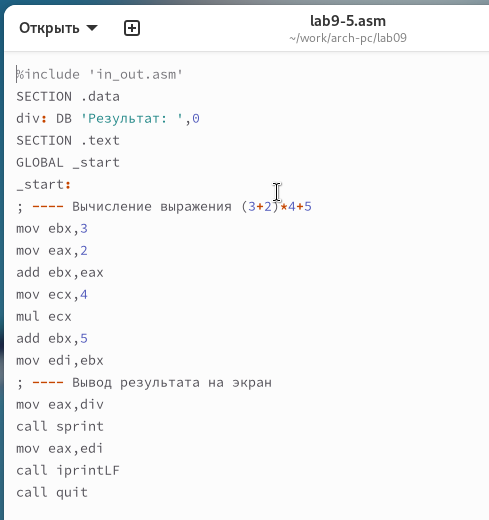


Рис. 18: Код с ошибкой

Заметим, что перепутан порядок аргументов у инструкции add и что по окончании работы в edi отправляется ebx вместо eax

Исправленный код программы (рис. **¿fig:019?**)

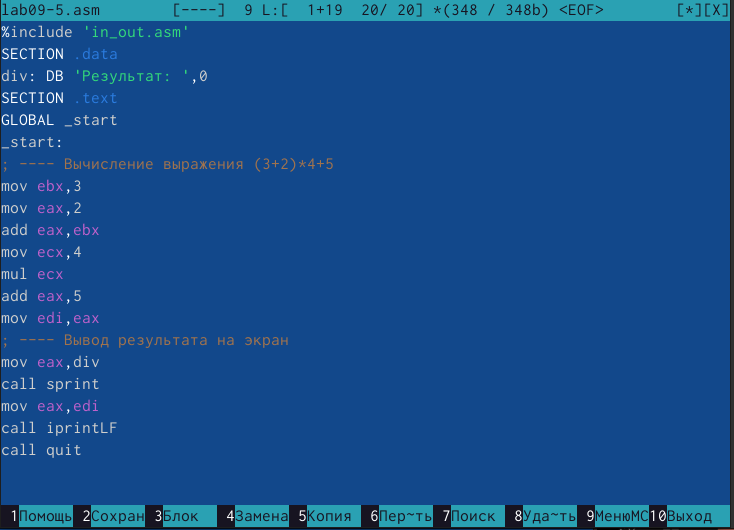


Рис. 19: Код исправлен

# 3 Выводы

Освоили работy с подпрограммами и отладчиком.