Лабораторная 9

Мазуркевич Анастасия

Содержание

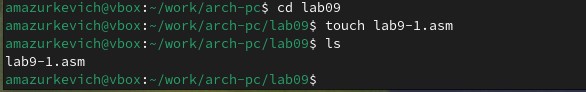
# Цель работы

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

# Выполнение лабораторной работы

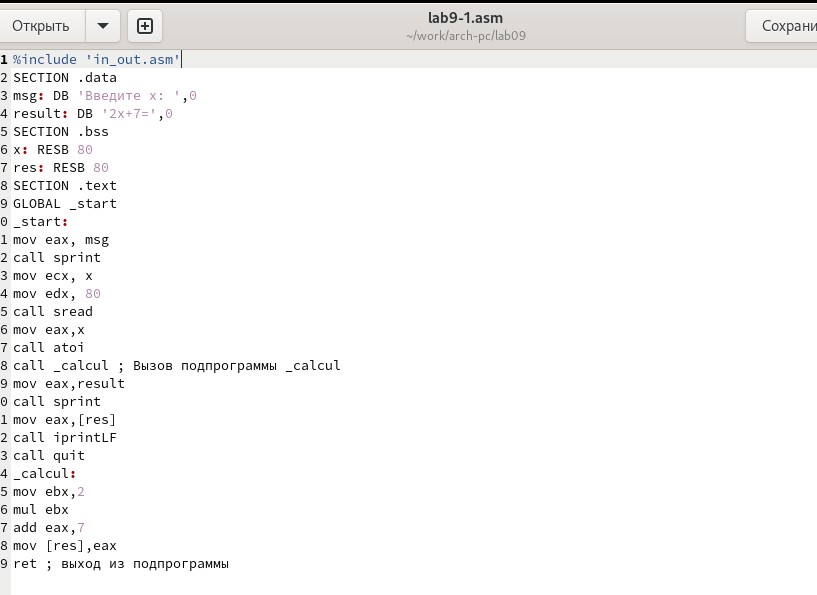
## Порядок выполнения лабораторной работы

Создайте каталог для выполнения лабораторной работы № 9, перейдите в него и создайте файл lab09-1.asm:



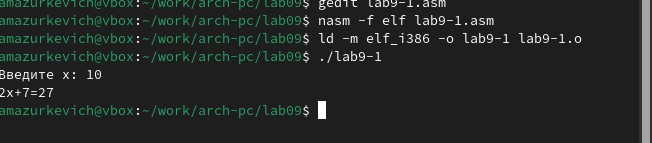
создаем каталог

Введите в файл lab9-1.asm текст программы из листинга 9.1



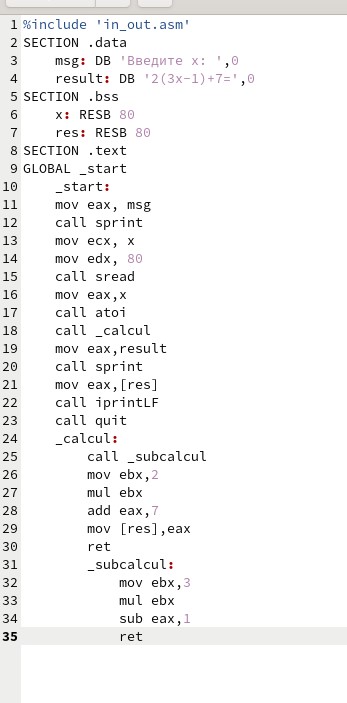
вводим

Создайте исполняемый файл и запустите его.



создаем и запускаем

Измените текст программы, добавив подпрограмму \_subcalcul в подпрограмму \_calcul, для вычисления выражения 𝑓(𝑔(𝑥)), где 𝑥 вводится с клавиатуры, 𝑓(𝑥) = 2𝑥 + 7, 𝑔(𝑥) = 3𝑥 − 1. Т.е. 𝑥 передается в подпрограмму \_calcul из нее в подпрограмму \_subcalcul, где вычисляется выражение 𝑔(𝑥), результат возвращается в \_calcul и вычисляется выражение 𝑓(𝑔(𝑥)). Результат возвращается в основную программу для вывода результата на экран.



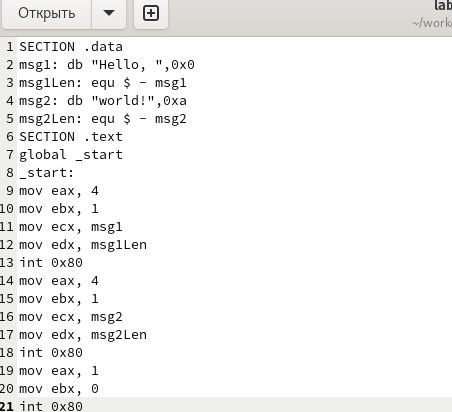
заменяем строки

Создайте исполняемый файл и запустите его.

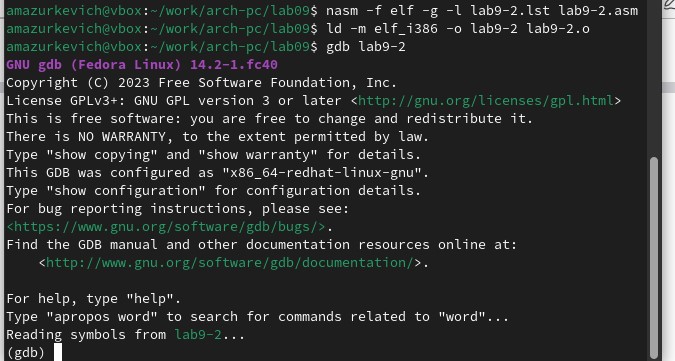


проверяем работу файла

Создайте файл lab09-2.asm с текстом программы из Листинга 9.2. (Программа печати сообщения Hello world!):

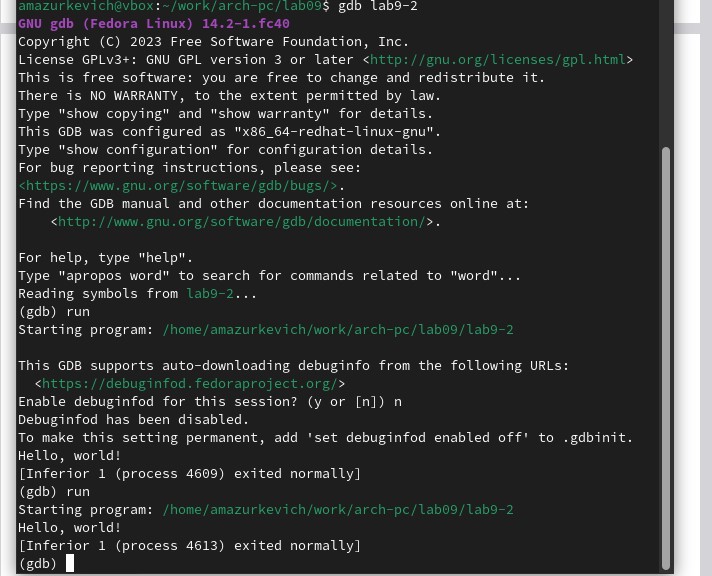


создаем



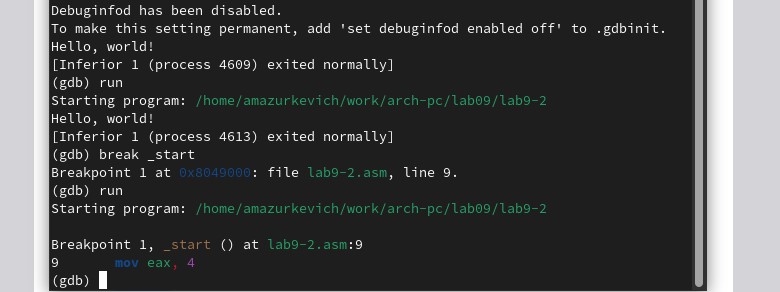
вводим код из листинга

Получите исполняемый файл.Для работы с GDB в исполняемый файл необходимо добавить отладочную информацию, для этого трансляцию программ необходимо проводить с ключом ‘-g’. Загрузите исполняемый файл в отладчик gdb:



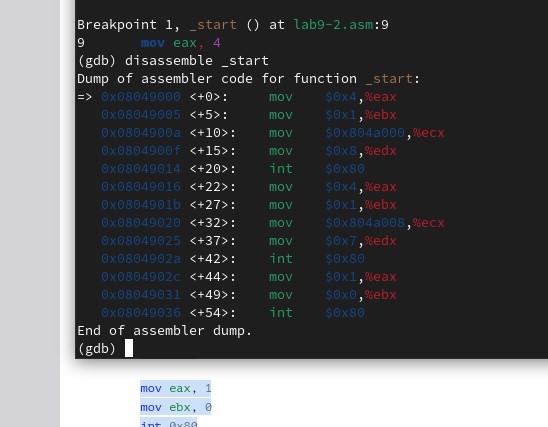
запускаем

Проверьте работу программы, запустив ее в оболочке GDB с помощью команды run (сокращённо r):



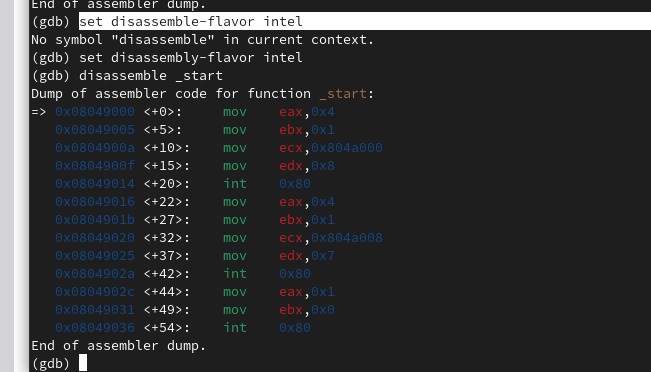
запускаем

Посмотрите дисассимилированный код программы с помощью команды disassemble начиная с метки \_start.



смотрим

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

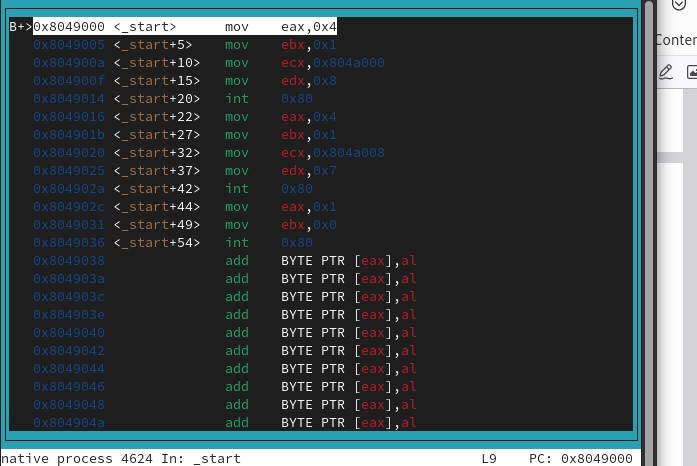


смотрим

Перечислите различия отображения синтаксиса машинных команд в режимах ATT и Intel.

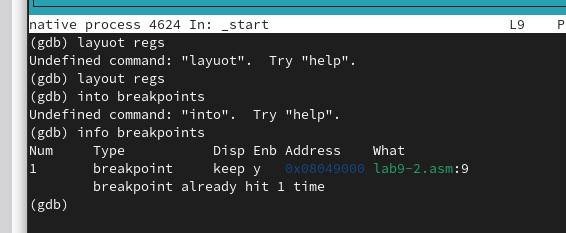
1. Порядок операндов
2. Синтаксис регистров
3. Синтаксис немедленных значений
4. Синтаксис адресов памяти
5. Команды для переходов и вызовов
6. Комментарии
7. Символы для указания на размер данных

Включите режим псевдографики для более удобного анализа программы (рис. 9.2):



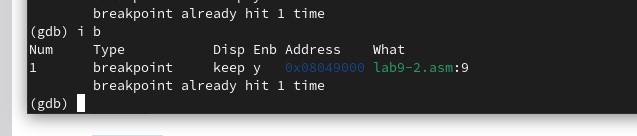
включаем

На предыдущих шагах была установлена точка останова по имени метки (\_start). Проверьте это с помощью команды info breakpoints (кратко i b)



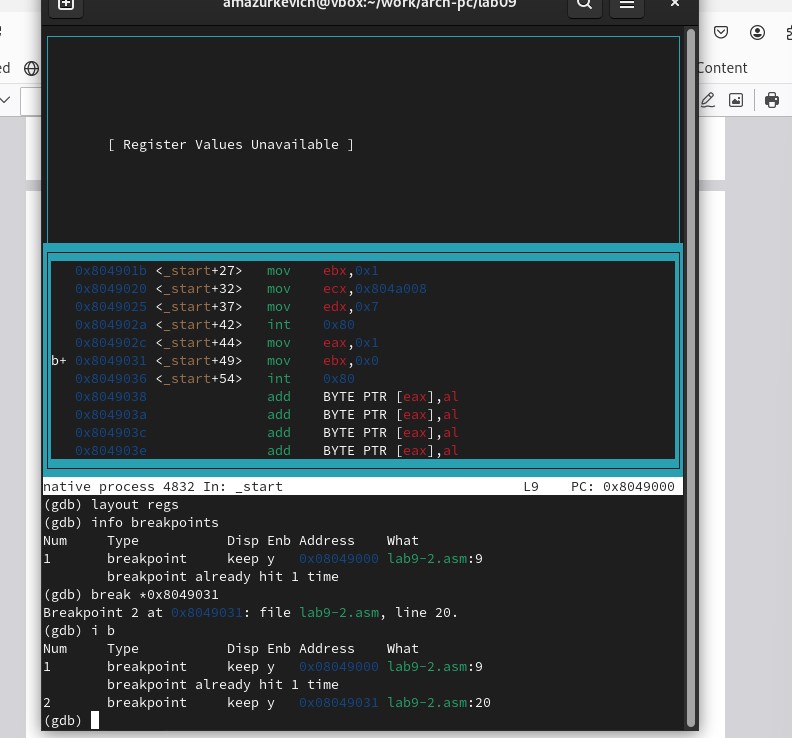
проверяем

Проверим с помощью i b



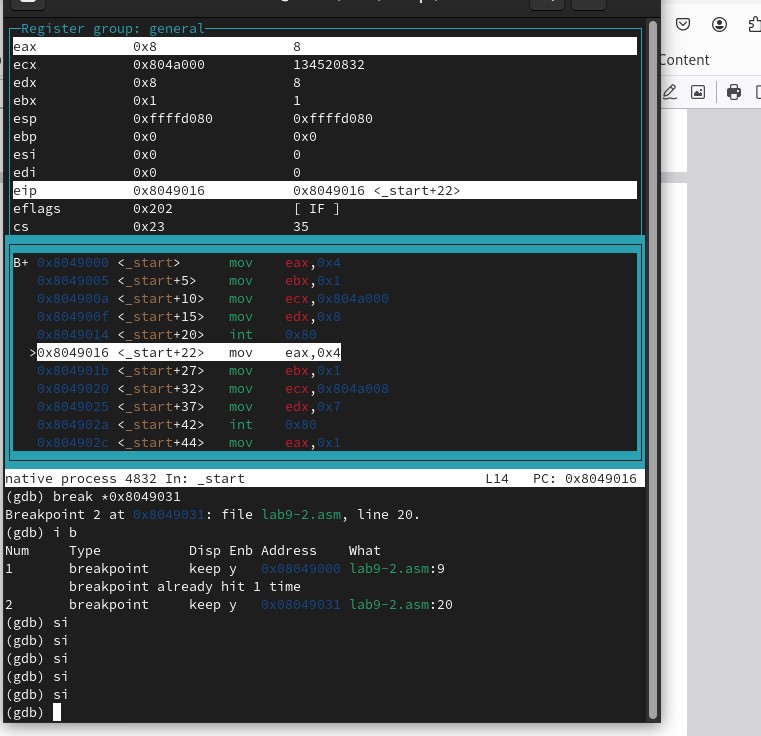
провеярем

Установим еще одну точку останова по адресу инструкции. Адрес инструкции можно увидеть в средней части экрана в левом столбце соответствующей инструкции (см. рис. 9.3). Определите адрес предпоследней инструкции (mov ebx,0x0) и установите точку останова.



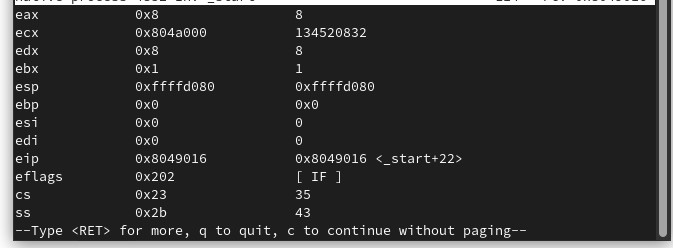
устанавливаем

Выполните 5 инструкций с помощью команды stepi (или si) и проследите за изменением значений регистров.



выполняем

Значения каких регистров изменяются? Во время выполнения команд менялись регистры: ebx, ecx, edx,eax, eip.



регистры

Смотрим msg1



смотрим

Смотрим msg1

смотрим

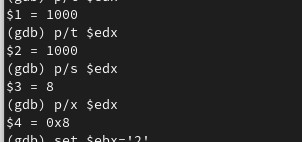
смотрим

Изменим первые символы



меняем

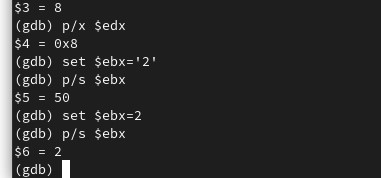
Чтобы посмотреть значения регистров используется команда print /F (перед именем регистра обязательно ставится префикс $) (рис. 9.6):



смотрим

изменяе ebx

Выводится разные значения команда без кавычек присваивает регистру вводимое значение.



меняем

завершаем программу и выходим



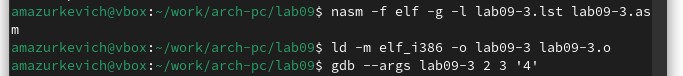
выходим

Скопируйте файл lab8-2.asm, созданный при выполнении лабораторной работы №8, с программой выводящей на экран аргументы командной строки (Листинг 8.2) в файл с именем lab09-3.asm:

копируем

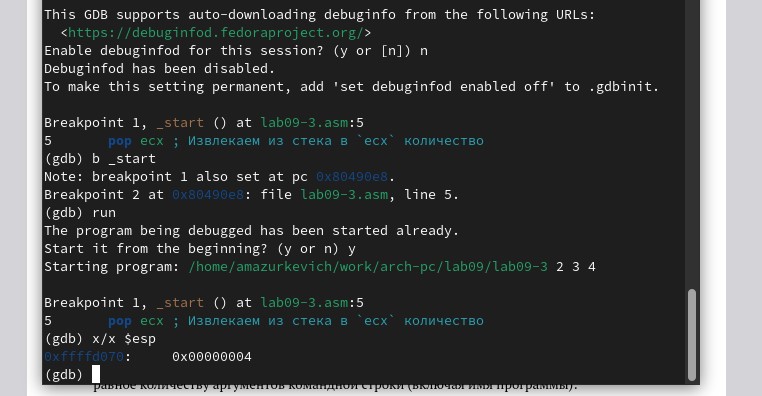
копируем

Создайте исполняемый файл



создаем

Для начала установим точку останова перед первой инструкцией в программе и запустим ее.



устанавливаем

Посмотрите 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.

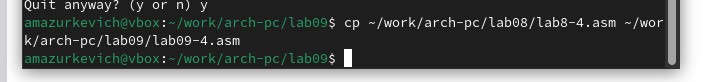


смотрим

Адресные регистры имеют размерность 32 бита, соответственно 4 байта

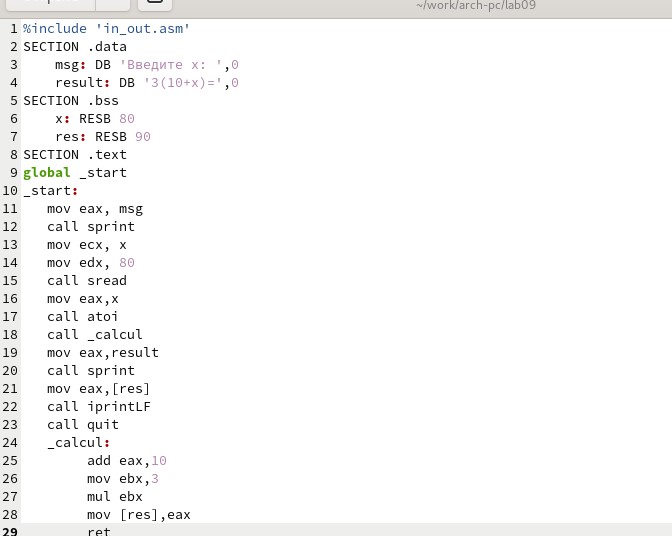
## Задание для самостоятельной работы

Преобразуйте программу из лабораторной работы №8 (Задание №1 для самостоятельной работы), реализовав вычисление значения функции 𝑓(𝑥) как подпрограмму



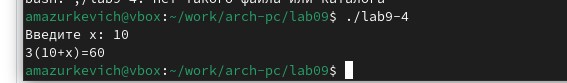
копируем файл

Вводим программу



пишем

Запускаем

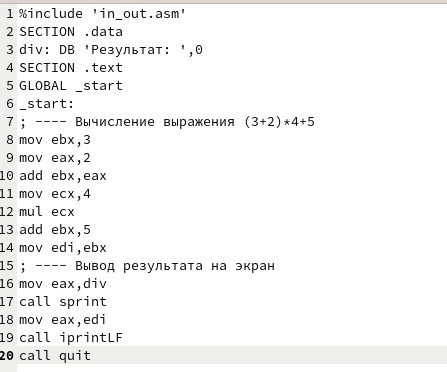


проверяем

В листинге 9.3 приведена программа вычисления выражения (3 + 2) ∗ 4 + 5. При запуске данная программа дает неверный результат. Проверьте это. С помощью отладчика GDB, анализируя изменения значений регистров, определите ошибку и исправьте ее.

создаем файл

создаем файл



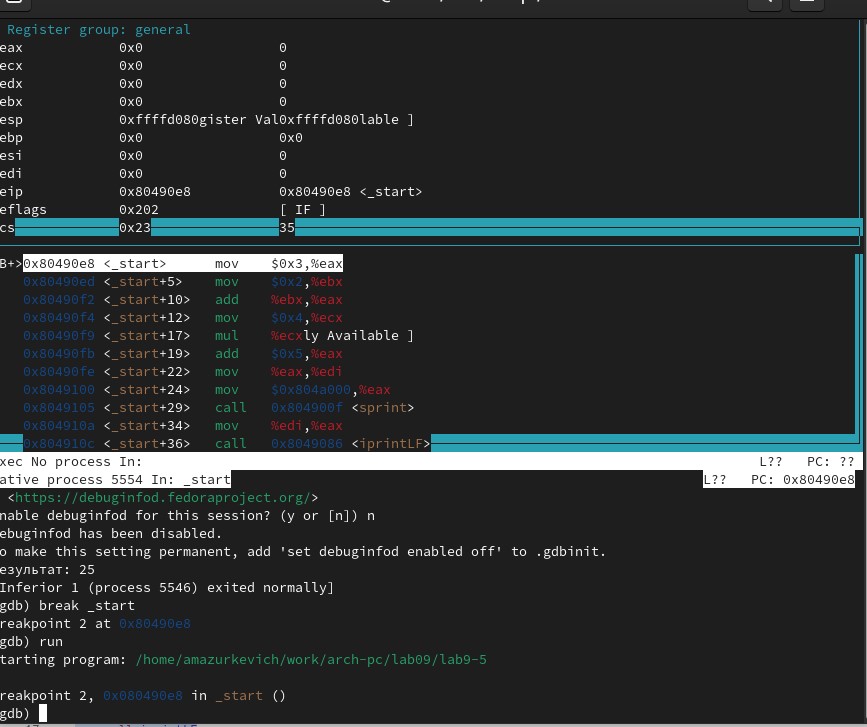
вводим листинг

Создаем файл и запускаем



запускаем

Запускаем его в отладчике GDB и смотрим на изменение решистров командой si



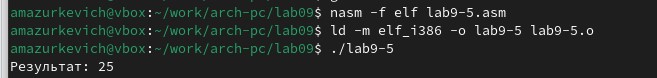
смотрим

Изменяем программу



меняем

Создаем исполняемый файл и запускаем его



проверяем

# Выводы

Приобрели навыки написания программ с использованием подпрограмм. Познакомились с методами отладки при помощи GDB и его основными возможностями.

# Список литературы