Лаборатрная работа №9

Понятие подпрограммы. Отладчик GDB.

Жукова Арина Александровна

Содержание

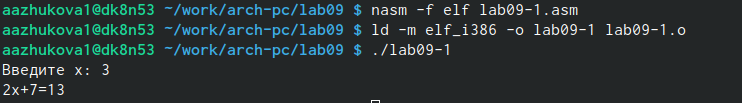
# 1 Цель работы

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

# 2 Выполнение лабораторной работы

## 2.1 Реализация подпрограмм в NASM

Создаём файл с текстом программы из листинга 9.1, создаём исполняеый файл и проверяем его работу (рис. ??).

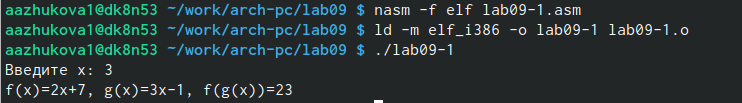


Проверка работы программы

Изменим текст программы, добавив подпрограмму \_subcalcul в подпрограмму \_calcul.

\_calcul:  
call \_subcalcul  
mov ebx,2  
mul ebx  
add eax,7  
mov [res],eax  
  
ret ; Выход из попрограммы  
;Подпрограмма для вычисления выражения "3x-1"  
\_subcalcul:  
mov ebx,3  
mul ebx  
sub eax,1  
mov [res],eax  
  
ret ; выход из подпрограммы

Проверим работу программы (рис. ??).



Проверка работы программы

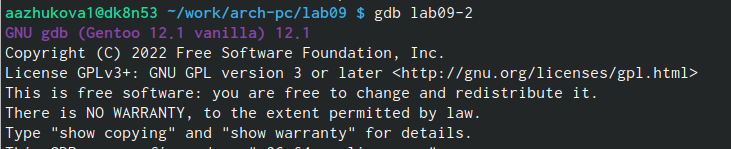
## 2.2 Отладка программам с помощью GDB

Создаём файл ab09-2.asm с текстом программы из Листинга 9.2. Получаем исполняемый файл, для работы с GDB добавляем отладочную информацию при помощи ключа ‘-g’ (рис. ??).

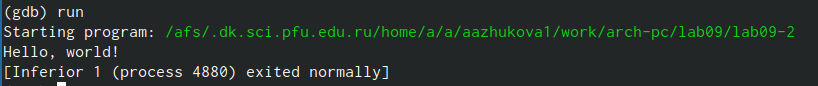
Получение исполняемого файла

Получение исполняемого файла

Загружаем исполняемый файл в отладчик gbd, проверяем работу программы в оболочке GBD при помощи команды run (рис. ??-??).

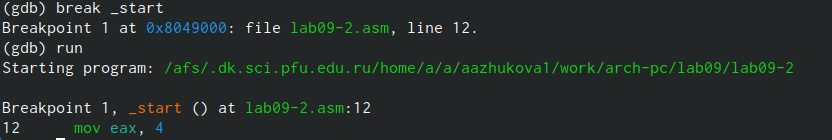


Загрузка файла в отладчик gbd



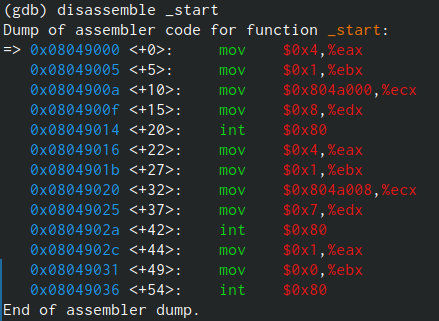
Проверка работы программы

Устанавливаем брейкпоинт на метку \_start, для более подробного анализа программы, и запускаем файл (рис. ??).



Запуск программ с брейкпоинтом

Просматриваем дисассимилированный код программы с помощью команды disassemble начиная с метки \_start (рис. ??).



Просмотр дисассимилированного кода

Переключаемся на отображение команд с Intel’овским синтаксисом (рис. ??).

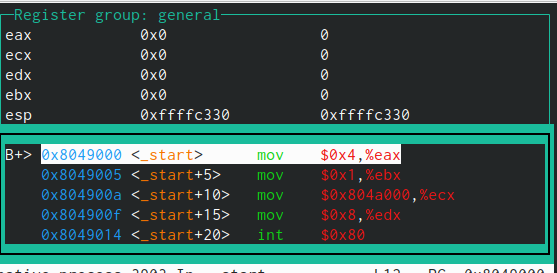


Отображение команд с Intel’овским синтаксисом

Intel’овский синтаксис отличается от синтаксиса ATT:

1. порядком операндов, в ATT сначала идёт источник затем приёмник;
2. имена регистров начинается с %, а название переменной с $;
3. числовые константы имеют знак $ в начале.

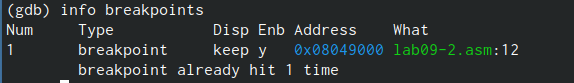
Включаем режим псевдографики для более удобного анализа программы, введя команды layout asm и layout regs (рис. ??).



Окна в режими псевдографики

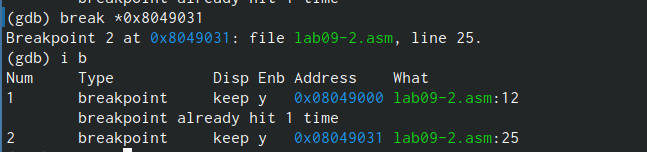
## 2.3 Добавление точек останова

Проверим установку точки останова с помощью команды info breakpoints (рис. ??).



Работа команды info breakpoints

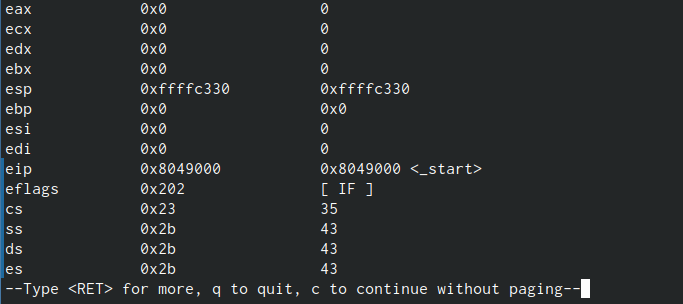
Установим ещё одну точку останова по адресу инструкции, просмотрим информацию по всем установленным точкам (рис. ??).



Информация о всех точках останова

## 2.4 Работа с данными программы в GDB

Просмотрим содержимое регистров при помощи команды info registers (рис. ??).



Содержимое регистров

Посмотрим значение переменной msg1 по имени (рис. ??).

Значение переменной по имени

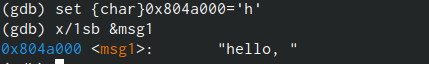
Значение переменной по имени

Посмотрим значение переменной msg2 по адресу (рис. ??).

Значение переменной по адресу

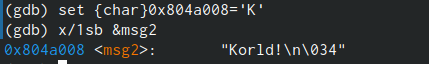
Значение переменной по адресу

Изменим значение переменной msg1 при помощи команды set (рис. ??).



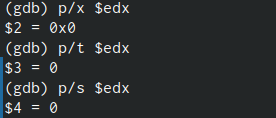
Замена значения переменной

Заменим первый символ во второй переменной msg2 на K (рис. ??).



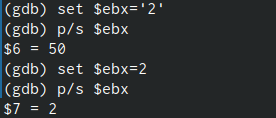
Замена символа

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



Значение регистра edx

Изменим значение регистра ebx при помощи команды set (рис. ??).

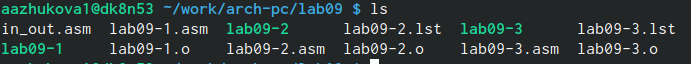


Изменение значения

В первом случае мы меняем значение регистра на символ ‘2’, и нам выводится значение 50 в соответствие с таблицей ASCII, а во втором знвчение меняется на цифру 2.

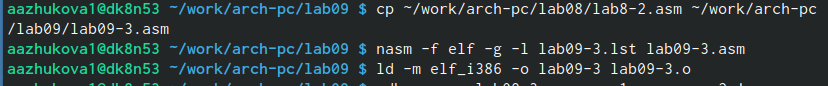
## 2.5 Обработка аргументов командной строки в GDB

Скопируем 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 (рис. ??).



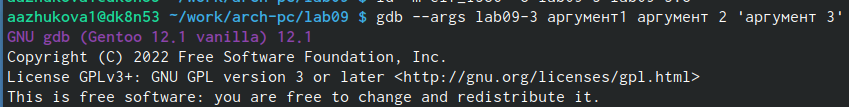
Изменение значения

Создадим исполняемый файл (рис. ??).



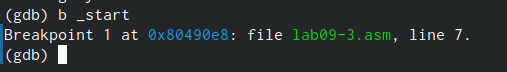
Создание исполняемого файла

Загрузим исполняемый файл в отладчик, указав аргументы, при помощи ключа –args (рис. ??).



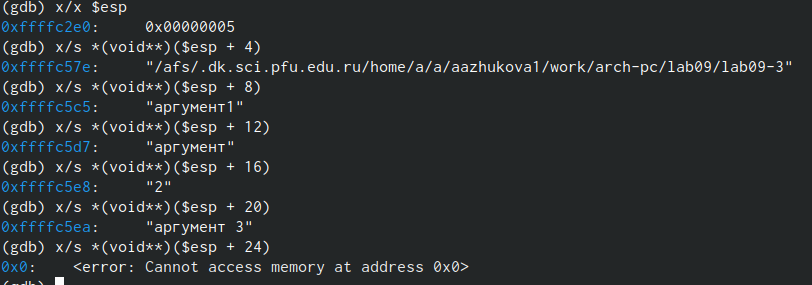
Загрузка исполняемого файла в отладчик

Установим точку останова перед первой инструкцией в программе и запустим ее (рис. ??).



Установка точки останова

Просмотрим позиции стека (рис. ??).



Позиции стека

Шаг изменения адреса равен 4 потому, что на каждое значение в памяти выделяется 4 байта.

# 3 Задания для самостоятельной работы

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

Копируем файл задания №1 для самостоятельной работы в папку с заданиями лабораторной работы под именем zadanie1.asm (рис. ??).

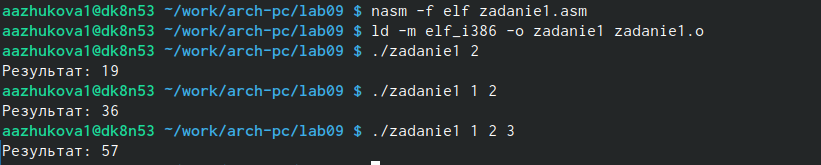
Проверка нахождения файла

Проверка нахождения файла

Изменяем текст программы так, чтобы вычисление значения функции f(x) вычислялась как подпрограмма (изменённая часть).

next:  
 cmp ecx, 0h  
 jz \_end  
 pop eax  
 call atoi  
 call \_calcul  
 loop next   
\_end:  
 mov eax,msg  
 call sprint  
 mov eax,esi  
 call iprintLF  
 call quit  
\_calcul:  
 mov ebx,2  
 mul ebx  
 add eax,15  
 add esi,eax  
 ret

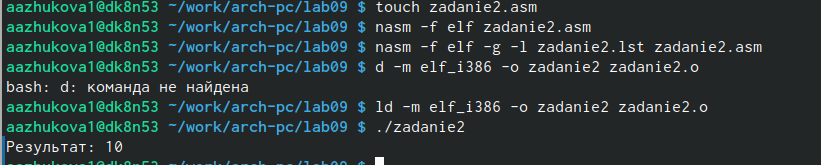
Создаём исполняемый файл и проверяем корректность работы программы (рис. ??).



Работа программы

1. 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.

Создаём файл zadanie2.asm и вносим в него текст листинга 9.3. Создаём исполняемый файл, добавляя в него отладоточную информацию, проверяем корректность работы программы. Выводиться неверный ответ (Результат: 10), верный (Результат: 29) (рис. ??).



Создание файла, проверка его работы

Исправим ошибку в прграмме с помощью отладчика GDB,анализируя изменения значений регистров

Правильная часть кода.

mov ebx,3   
mov eax,2  
add ebx,eax  
mov eax,ebx  
mov ecx,4  
mul ecx  
add eax,5  
mov edi,eax

Проверим корректность работы программы (рис. ??)

Работа программы

Работа программы

# 4 Выводы

В ходе выполнения лабораторной работы мы приобрели навыки написания программ с использованием подпрограмм, познакомтлтсб с методами отладки при помощи GDB и его основными возможностями.

# Список литературы