Отчёт по лабараторной работе №9

Понятие подпрограммы. Отладчик GDB.

Корчаги Алексей Павлович

Содержание

# 1 Цель работы

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

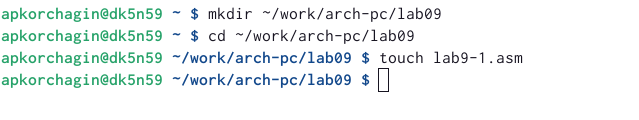
# 2 Теоретическое введение

Отладка — это процесс поиска и исправления ошибок в программе. В общем случае его можно разделить на три этапа: • посик ошибки; • определение причины ошибки; • исправление ошибки.

GDB — отладчик проекта GNU работает на большенстве UNIX-подобных системах и может проводить процесс отладки для кода написанного на многих языках программирования. GDB предлагает обширные средства для слежения и контроля за выполнением компьютерных программ. От- ладчик не содержит собственного графического пользовательского интерфейса и использует стандартный текстовый интерфейс консоли. Однако для GDB существует несколько сторон- них графических надстроек, а кроме того, некоторые интегрированные среды разработки используют его в качестве базовой подсистемы отладки. Отладчик GDB (как и любой другой отладчик) позволяет увидеть, что происходит «внутри» программы в момент её выполнения или что делает программа в момент сбоя. GDB может выполнять следующие действия: • начать выполнение программы, задав всё, что может повлиять на её поведение; • остановить программу при указанных условиях; • исследовать, что случилось, когда программа остановилась

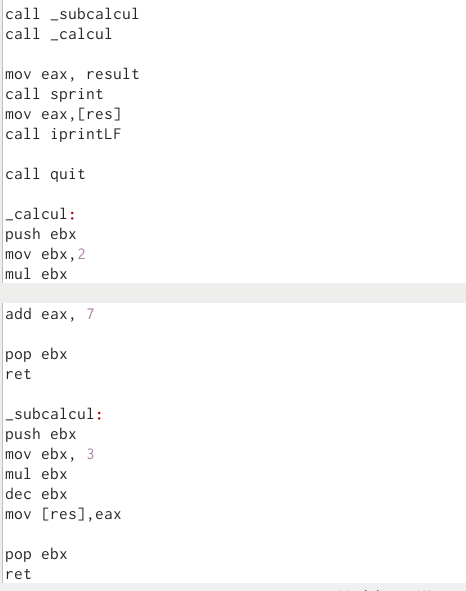
# 3 Выполнение лабораторной работы

Создал директорию и файл для выполненние лаборотной работы(рис. ??).



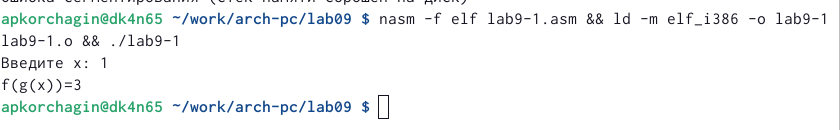
Созданние файла

Записал код в файл lab9-1.asm, дописал доп функцию для того чтобы программа работала как сложная функции(рис. ??).



Редактированние файла

Проверил работу программы(рис. ??).



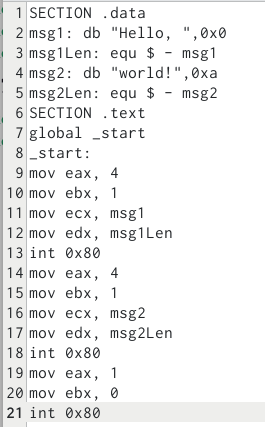
Работа программы

Создал файл lab9-2.asm(рис. ??).

Созданние файла

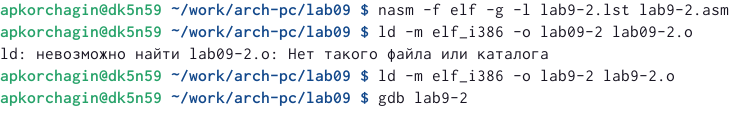
Созданние файла

Записал код в файл(рис. ??).



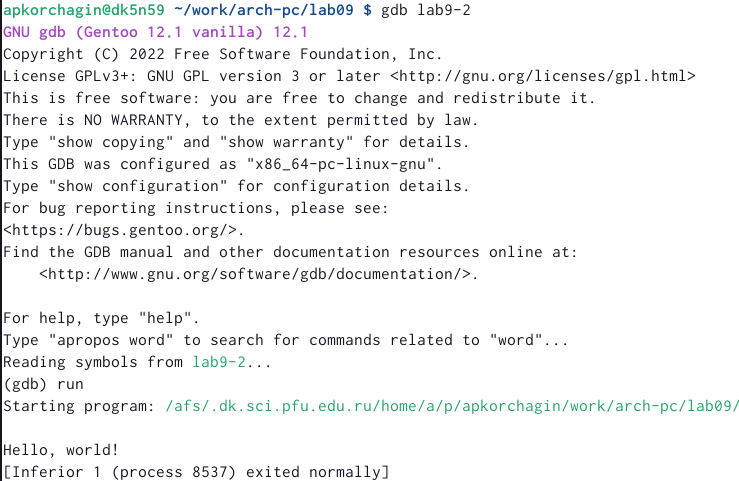
Редактированние файла

Ассемблирол файл lab9-2.asm, чтобы открыть файл через gdb(рис. ??).



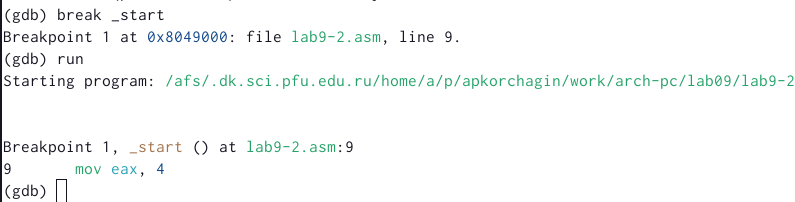
Работа файла

Открыл файл lab9-2.asm через gdb(рис. ??).



Работа с gdb

Поставил breakpoint на \_start(рис. ??).



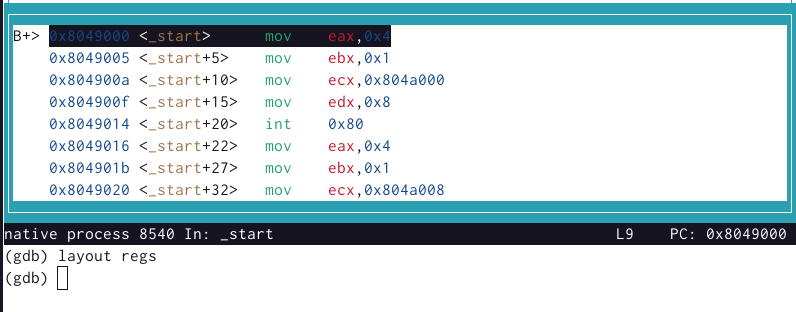
Установка breakpoint

Начал просмотр дизассемблированного кода с синтаксисом intel(рис. ??).



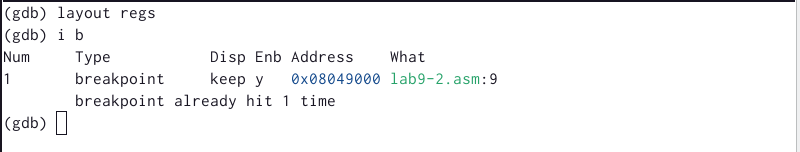
Дизассемблированный код

В синтаксисе ATT в виде 16-ти ричного числа записаны первые аргументы всех комманд, в синтаксисе Intel так записываются адреса вторых аргументов. запустим режим псевдографики, с помощью которого отображается код программы и содержимое регистров(рис. ??).



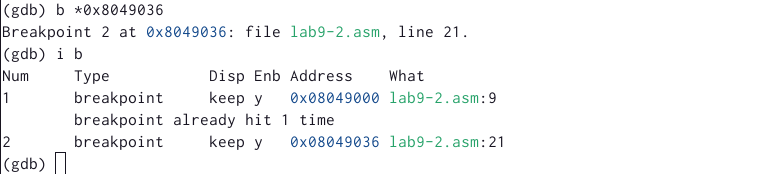
Содержанние регистров

С помощью i b посмотрим информацию о точках остановки(рис. ??).



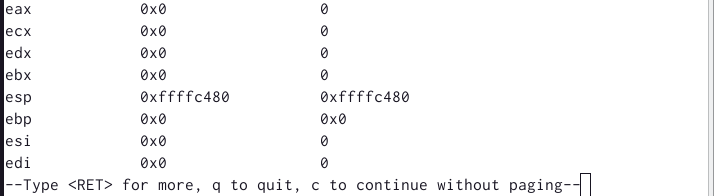
Команда i b

Добавил ещё одну точку остановки по адресу(рис. ??).



Взаимодействие с адресом

С помощью команды i r вывел значенние регистров(рис. ??).



Значенние регистров

Вывел значение переменной по именни(рис. ??).



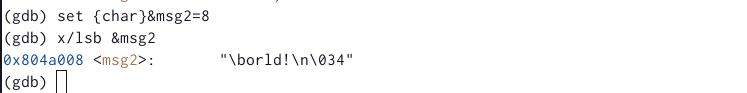
Значенние переменной msg1

Изменил значенние переменной msg1(рис. ??).



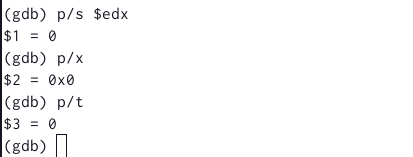
Измененние значения msg1

Изменил значенние переменной msg2(рис. ??).



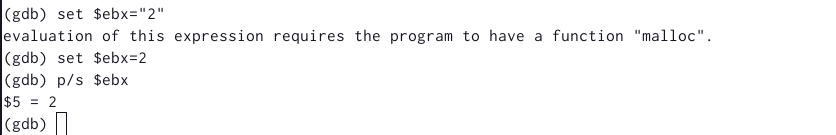
Измененние значения msg2

Вывел значение регистра ebx, в трёх разных форматах: в строчном(p/s); в 16-речном(p/x);в двоичном(p/t)(рис. ??).



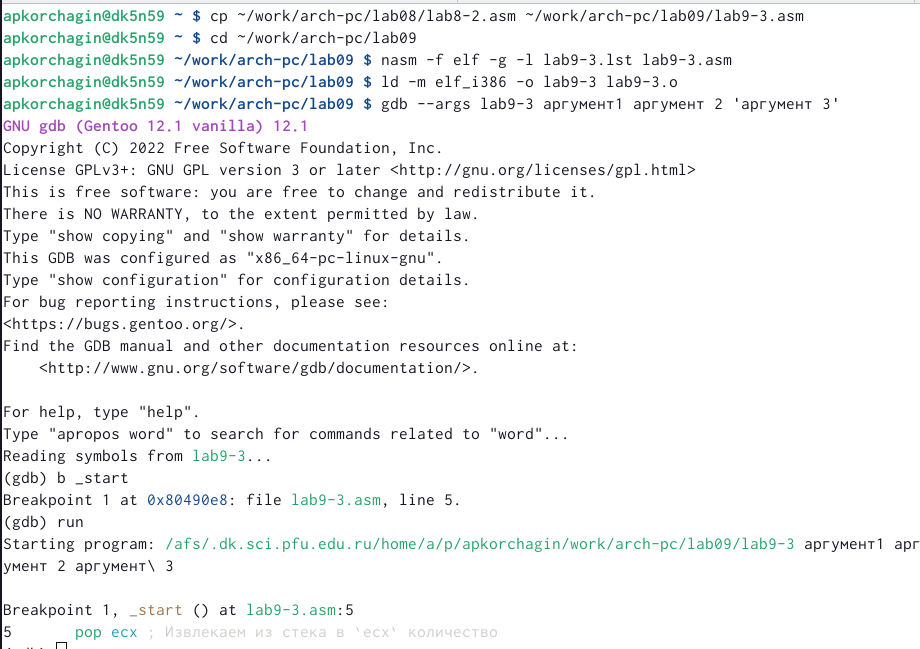
Значенние регистра

Задал значенние регистру, при попытке задания регустру строкового занченния возникает ошибка(рис. ??).



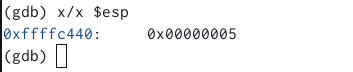
Присваивание занченния регистру

Скопировал из 8 лабороторной работы файл переименовал и создал исполняймый файл. Открыл файл создал точку остановки(рис. ??).



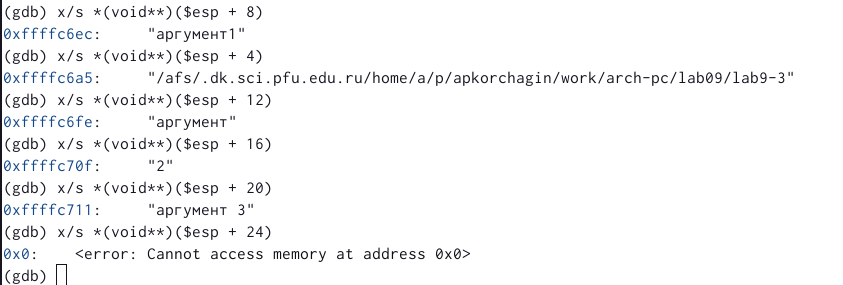
Копирование файла

Посмотерл на содержимое того, что находится по адресу регистра esp(рис. ??).



Значенние esp

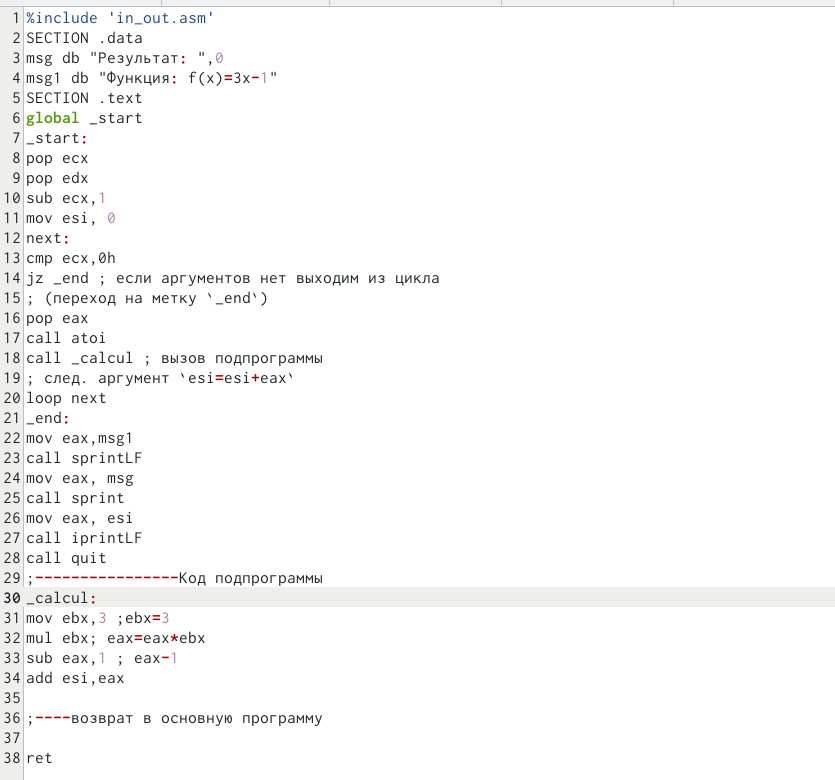
Посмотрел на все остальные элементы стека. Их адреса находятся на растоянии 4 байта, именно столько занимает один элемент стека(рис. ??).



Элементы стека

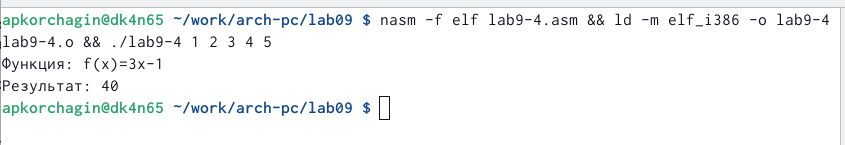
# 4 Выполенние самостоятельной работы

Скопировал и переименновал файл, переписал программу с использованием под подпрограм(рис. ??).



Редактированние файла

Создал исполняймый файл и проверил работу изменённой программы(рис. ??).

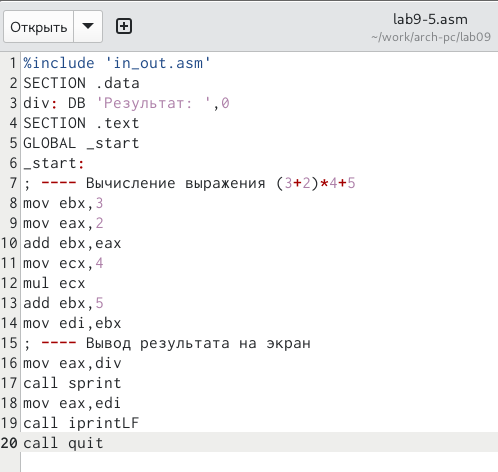


Созданние исполняймого файла

Код программы задания 1:

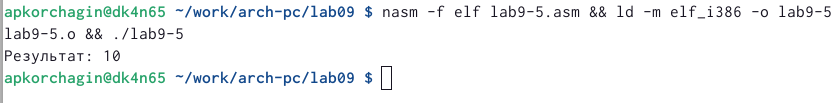
%include 'in\_out.asm'  
SECTION .data  
msg db "Результат: ",0  
msg1 db "Функция: f(x)=3x-1"  
SECTION .text  
global \_start  
\_start:  
pop ecx   
pop edx   
sub ecx,1   
mov esi, 0   
next:  
cmp ecx,0h   
jz \_end ; если аргументов нет выходим из цикла  
; (переход на метку `\_end`)  
pop eax   
call atoi   
call \_calcul ; вызов подпрограммы  
; след. аргумент `esi=esi+eax`  
loop next   
\_end:  
mov eax,msg1   
call sprintLF   
mov eax, msg   
call sprint  
mov eax, esi   
call iprintLF   
call quit   
;----------------Код подпрограммы  
\_calcul:  
mov ebx,3 ;ebx=3  
mul ebx; eax=eax\*ebx  
sub eax,1 ; eax-1  
add esi,eax   
  
;----возврат в основную программу  
  
ret

Создал файл lab9-5.asm и ввёл в него код из листинга(рис. ??).



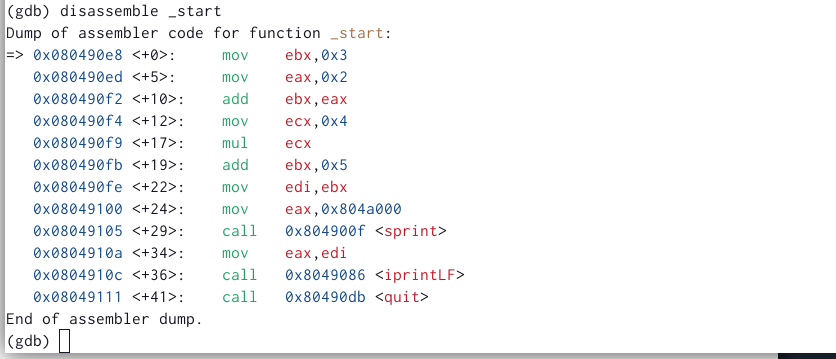
Программа из листинга

Создал исполняймый файл и запустил программу, программа выводит не верный ответ(рис. ??).



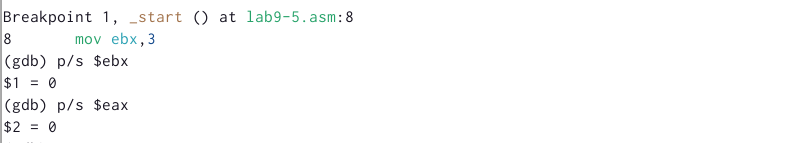
Созданние исполняймого файла

Открыл прогрмму в отладчике. Для того, чтобы найти ошибку, смотрю дисассемблирую программу и добавляю брейкпоинты в основные части программы (рис. ??).



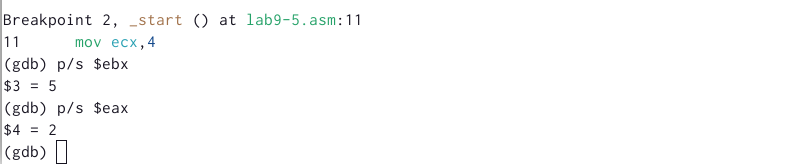
Отладчик

Перешёл к первому брекпоинту, в точке \_start, ошибок не обнаружил(рис. ??).



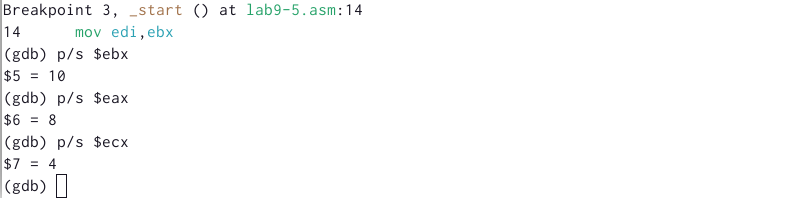
Первая точка остановки

Перешёл к второй точке остановки, заметил, что результат суммирования записывается в ebx (рис. ??).



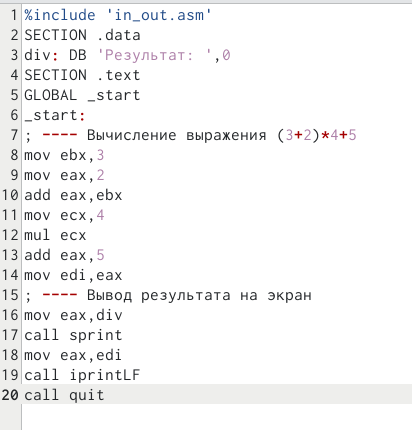
Вторая точка остановки

Перешёл к третьему брекпоинту, проверил значение регистров, заметил что умножение значения регистра ecx происходит на регистр eax(4\*2), а к регистру ebx прибовляется 5 (5+5) и его значенние записыватся в результат программы. (рис. ??).



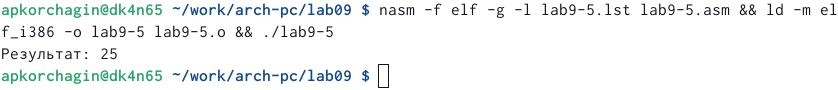
Третья точка остановки

Исправил ошибки, которые нашй благодаря GDB(рис. ??).



Редактированние файла

Создал исполняймый файл и проверил программу, теперь программа выводить коректное значенние(рис. ??).



Созданние исполняймого файла

Код программы задания 2:

%include 'in\_out.asm'  
SECTION .data  
div: DB 'Результат: ',0  
SECTION .text  
GLOBAL \_start  
\_start:  
; ---- Вычисление выражения (3+2)\*4+5  
mov ebx,3  
mov eax,2  
add eax,ebx  
mov ecx,4  
mul ecx  
add eax,5  
mov edi,eax  
; ---- Вывод результата на экран  
mov eax,div  
call sprint  
mov eax,edi  
call iprintLF  
call quit

# 5 Выводы

В ходе выполненния работа я приобрёл навыки написания программ с использованием подпрограмм и познакомился с методами отладки при помощи GDB.