Лабораторная работа №9

Понятие подпрограммы. Отладчик GDB.

Солдатов Алексей

Содержание

# 1 Цель работы

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

# 2 Задание

1. Реализация подпрограмм в NASM
2. Отладка программ с помощью GDB
3. Выполнение заданий для самостоятельной работы

# 3 Теоретическое введение

Здесь описываются теоретические аспекты, связанные с выполнением работы.

Например, в табл. [1](#tbl:std-dir) приведено краткое описание стандартных каталогов Unix.

Table 1: Описание некоторых каталогов файловой системы GNU Linux

| Имя каталога | Описание каталога |
| --- | --- |
| / | Корневая директория, содержащая всю файловую |
| /bin | Основные системные утилиты, необходимые как в однопользовательском режиме, так и при обычной работе всем пользователям |
| /etc | Общесистемные конфигурационные файлы и файлы конфигурации установленных программ |
| /home | Содержит домашние директории пользователей, которые, в свою очередь, содержат персональные настройки и данные пользователя |
| /media | Точки монтирования для сменных носителей |
| /root | Домашняя директория пользователя root |
| /tmp | Временные файлы |
| /usr | Вторичная иерархия для данных пользователя |

Более подробно об Unix см. в [1–6].

# 4 Выполнение лабораторной работы

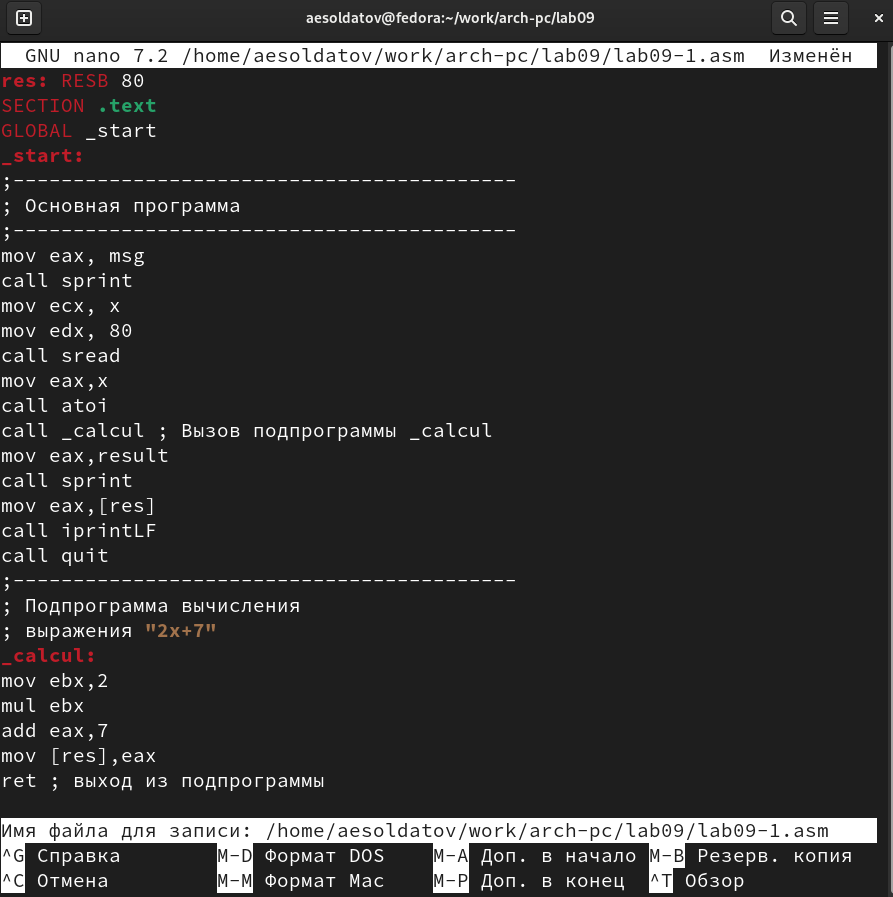
## 4.1 Реализация подпрограмм в NASM

Создал каталог для программ лабораторной работы №9, перешел в него и создал файл “lab9-1.asm” (рис. ??).



Подготовка к работе

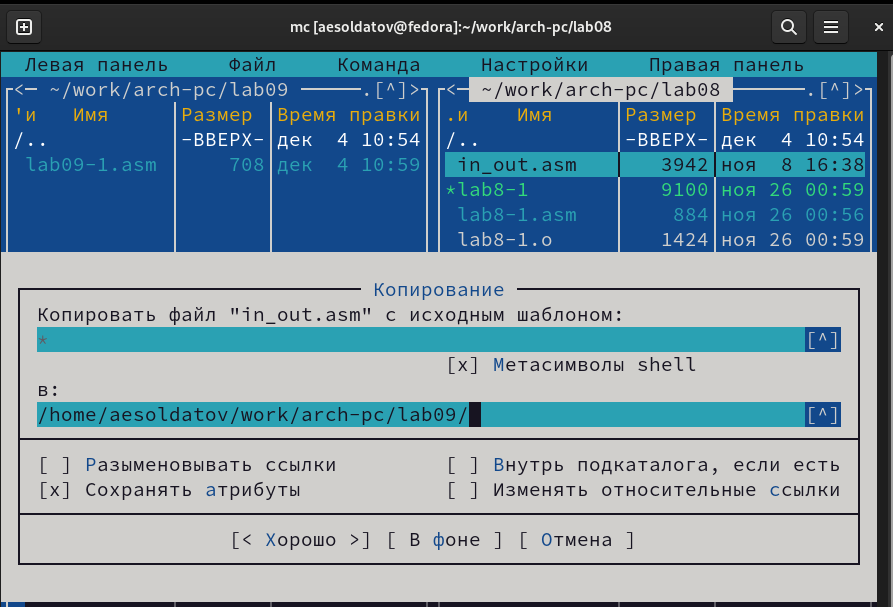
Внимательно изучил текст программы из листинга 9.1 со страницы в ТУИС и ввел в файл “lab9-1.asm” текст программы (рис. ??).



Ввод команд

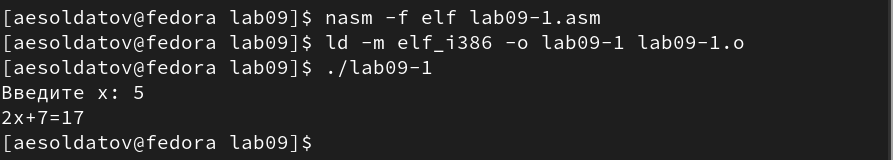
%include 'in\_out.asm  
SECTION .data  
msg: DB 'Введите x: ',0  
result: DB '2x+7=',0  
SECTION .bss  
x: RESB 80  
res: RESB 80  
SECTION .text  
GLOBAL \_start  
\_start:  
;------------------------------------------  
; Основная программа  
;------------------------------------------  
mov eax, msg  
call sprint  
mov ecx, x  
mov edx, 80  
call sread  
mov eax,x  
call atoi  
call \_calcul ; Вызов подпрограммы \_calcul  
mov eax,result  
call sprint  
mov eax,[res]  
call iprintLF  
call quit  
;------------------------------------------  
; Подпрограмма вычисления  
; выражения "2x+7"  
\_calcul:  
mov ebx,2  
mul ebx  
add eax,7  
mov [res],eax  
ret ; выход из подпрограммы

Перенес файл “in\_out.asm” из прошлой папки с лабораторной работой в нынешнюю (рис. ??).



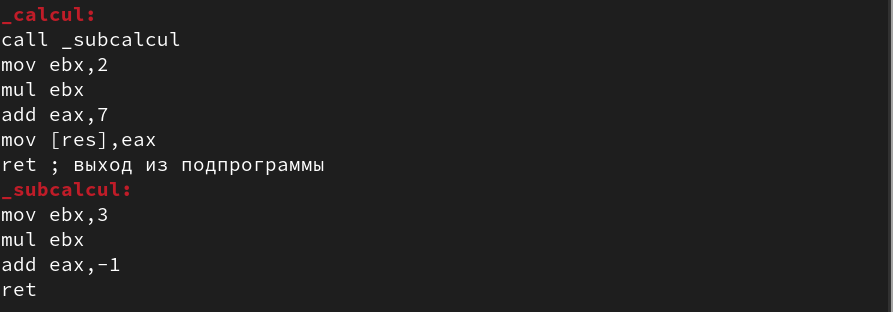
Перенос файла

Создал исполняемый файл и запустил его (рис. ??).



Создание и запуск файла

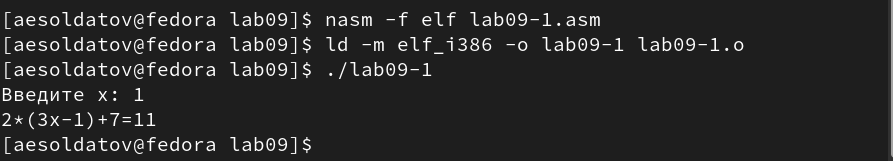
Далее изменил текст программы добавив подпрограмму “\_subcalcul” в подпрограмму “\_calcul” согласно заданию (рис. ??).



Изменение текста программы

%include 'in\_out.asm'  
SECTION .data  
msg: DB 'Введите x: ',0  
result: DB '2\*(3x-1)+7=',0  
SECTION .bss  
x: RESB 80  
res: RESB 80  
SECTION .text  
GLOBAL \_start  
\_start:  
;------------------------------------------  
; Основная программа  
;------------------------------------------  
mov eax, msg  
call sprint  
mov ecx, x  
mov edx, 80  
call sread  
mov eax,x  
call atoi  
call \_calcul ; Вызов подпрограммы \_calcul  
mov eax,result  
call sprint  
mov eax,[res]  
call iprintLF  
call quit  
;------------------------------------------  
; Подпрограмма вычисления  
; выражения "2x+7"\_calcul:  
call \_subcalcul  
mov ebx,2  
mul ebx  
add eax,7  
mov [res],eax  
ret ; выход из подпрограммы  
\_subcalcul:  
mov ebx,3  
mul ebx  
add eax,-1  
ret

Создал исполняемый файл и запустил его, все работает как надо (рис. ??).



Создание и проверка работы файла

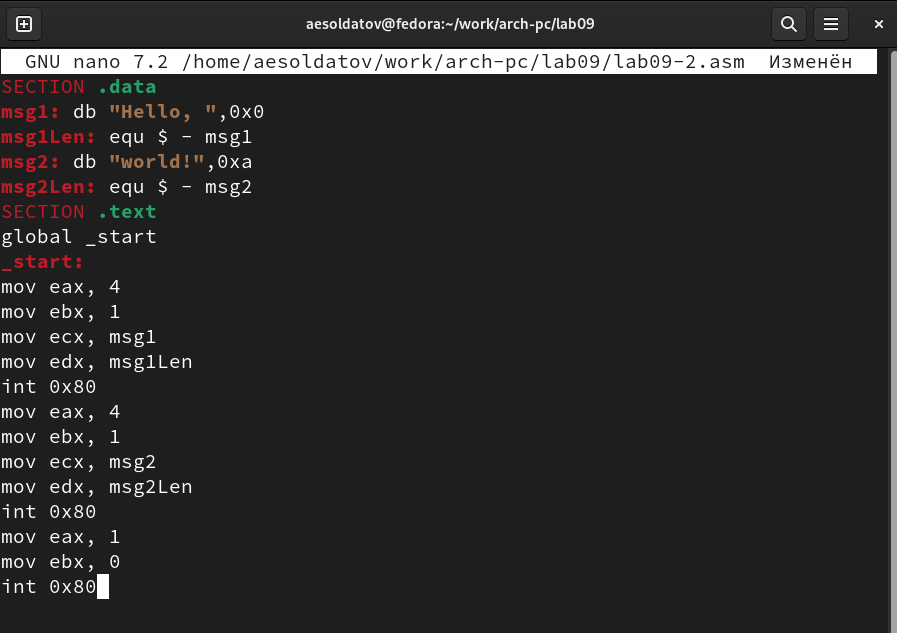
## 4.2 Отладка программ с помощью GDB

Создал файл “lab9-2.asm” в каталоге “~/work/arch-pc/lab09” (рис. ??).

Создание файла

Создание файла

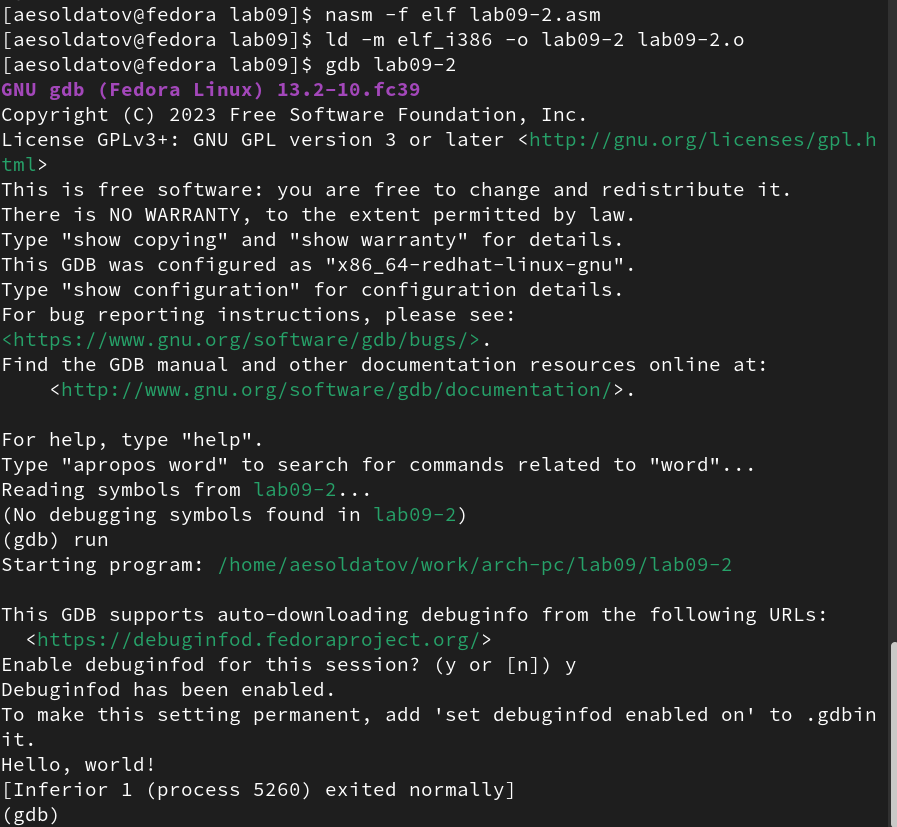
Внимательно изучил текст программы из листинга 9.2 и ввел его в файл (рис. ??).



Ввод команд

SECTION .data  
msg1: db "Hello, ",0x0  
msg1Len: equ $ - msg1  
msg2: db "world!",0xa  
msg2Len: equ $ - msg2  
SECTION .text  
global \_start  
\_start:  
mov eax, 4  
mov ebx, 1  
mov ecx, msg1  
mov edx, msg1Len  
int 0x80  
mov eax, 4  
mov ebx, 1  
mov ecx, msg2  
mov edx, msg2Len  
int 0x80  
mov eax, 1  
mov ebx, 0  
int 0x80

Создал исполняемый файл с отладочной информацией, загрузил его в отладчик gdb и проверил его работу, запустив в оболочке gdb, с помощью команды “run”. Программа вывела “Hello, world!” (рис. ??).



Работа с файлом

Для более подробного анализа программы установил брейкпоинт на метку “\_start” и запустил отладку (рис. ??).



Установка точки останова

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



Дисассимилированный код программы

Переключился на отображение команд с Intel’овским синтаксисом, введя команду “set disassembly-flavor intel” (рис. ??).

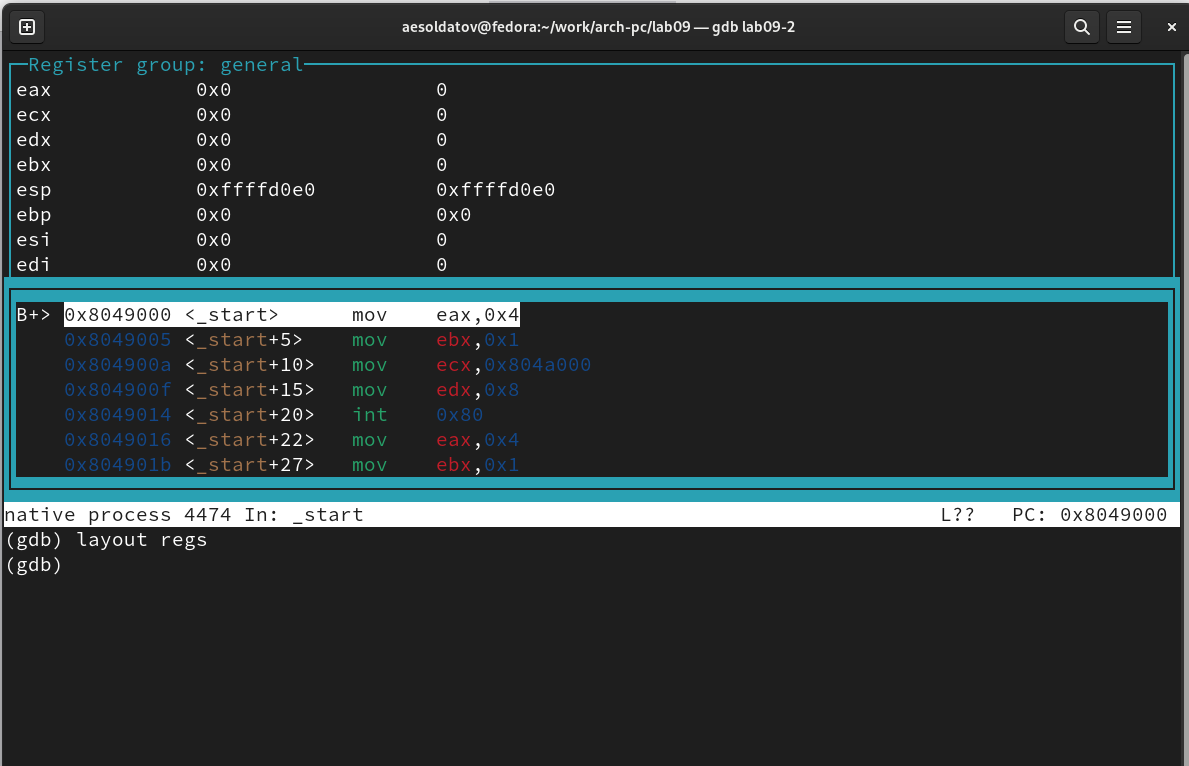


Отображение команд с Intel’овским синтаксисом

Различия отображения синтаксиса машинных команд в режимах ATT и Intel:

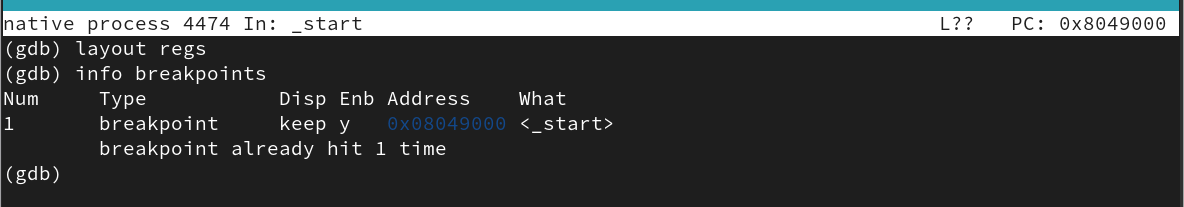
1. Порядок операндов (в режиме ATT начала источник, потом приемник, а в Intel сначала приемник, потом источник)
2. Отсутствие специальных символов ($, %) в Intel

Включил режим псевдографики для более удобного анализа программы (рис. ??).



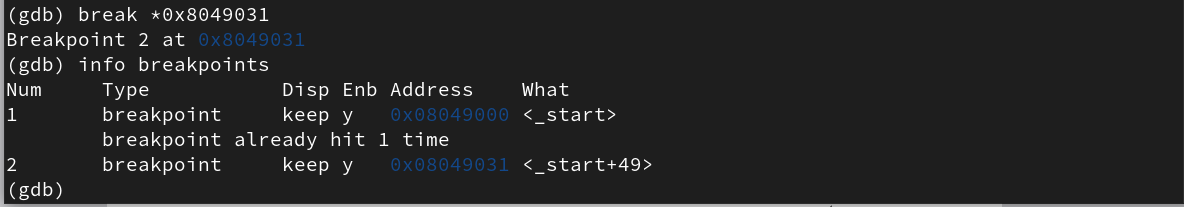
Режим псевдографики

Проверил работу команды “info breakpoints” (рис. ??).



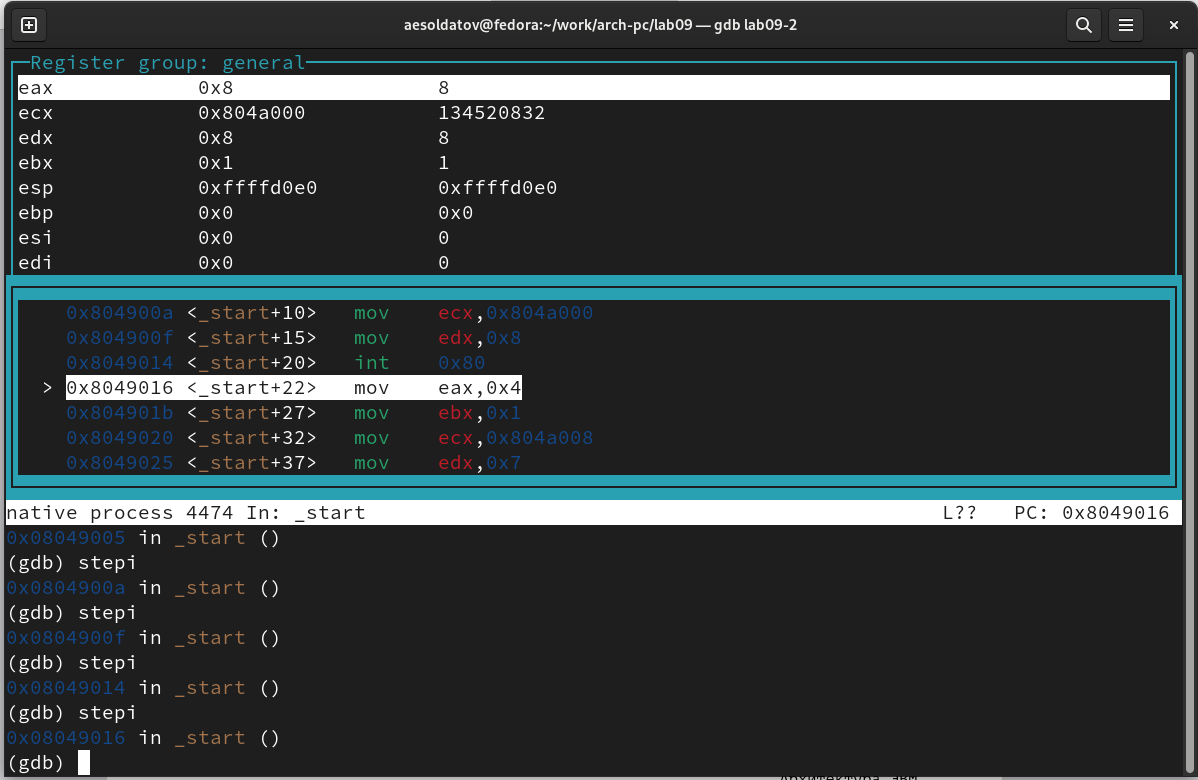
info breakpoints

Установил еще одну точку останова по адресу инструкции и проверил информацию о всех установленных точках останова (рис. ??).



Новая точка останова

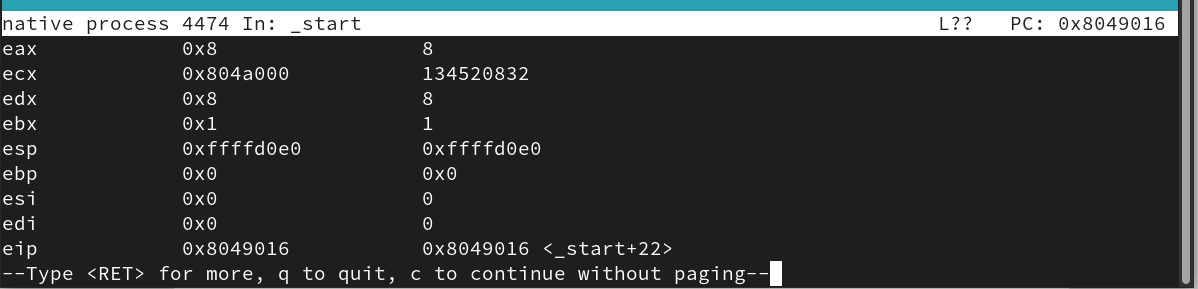
Выполнил 5 инструкций с помощью команды “stepi” и проследил за изменением значений регистров (рис. ??).



Работа “stepi”

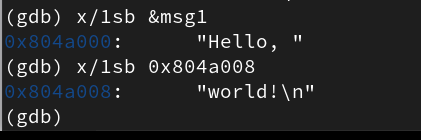
Изменились значения регистров eax, ebx, ecx, edx

Посмотрел содержимое регистров с помощью команды “info registers” (рис. ??).



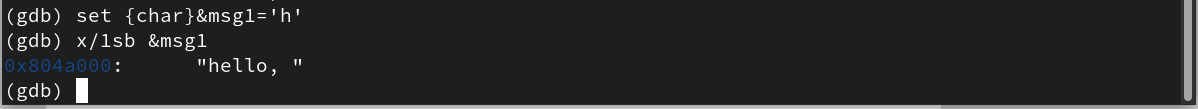
info registers

Посмотрите значение переменной “msg1” по имени и посмотрел значение переменной “msg2” по адресу. Адрес переменной определил по дизассемблированной инструкции (рис. ??).



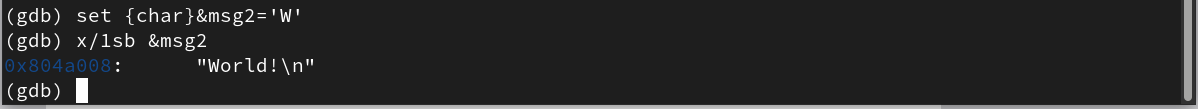
Значения msg1 и msg2

Изменил первый символ переменной “msg1” (рис. ??).



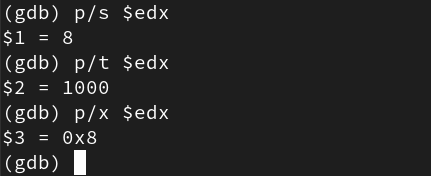
Изменение msg1

Изменил первый символ переменной “msg2” (рис. ??).



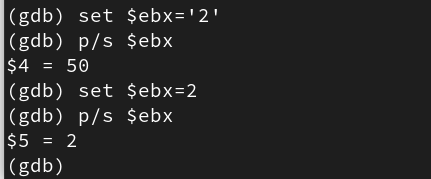
Изменение msg2

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



Значение регистра edx

С помощью команды “set” измените значение регистра ebx (рис. ??).

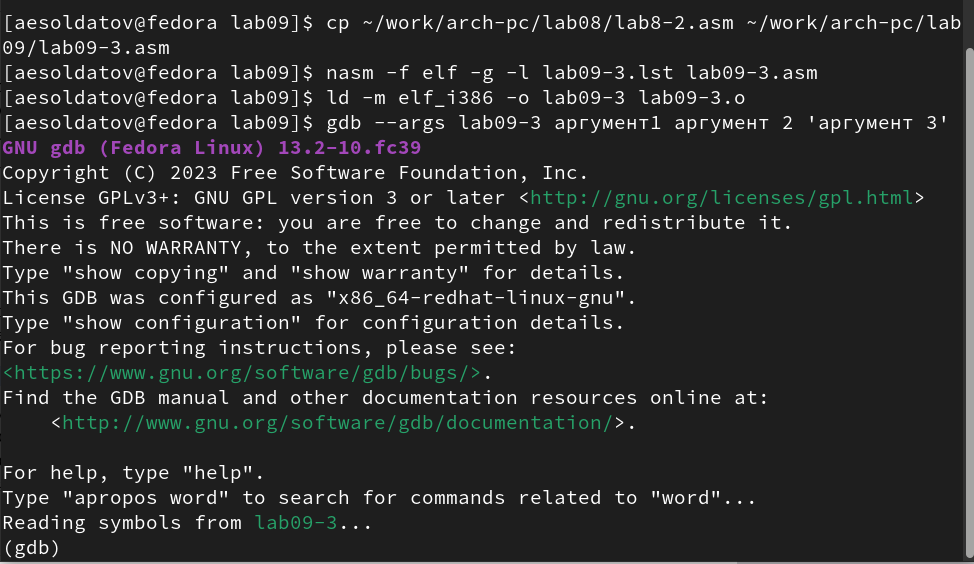


Изменение регистра edx

Значение регистра отличаются,так как в первом случае мы выводим код символа 2, который в десятичной системе счисления равен 50, а во втором случае выводится число 2, представленное в этой же системе.

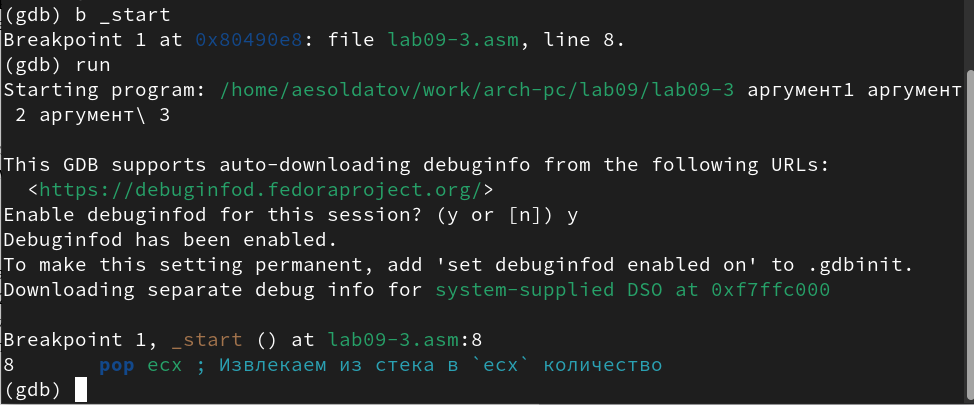
Завершаю выполнение программы с помощью команды “continue” и выхожу из GDB с помощью команды “quit”

Скопировал файл lab8-2.asm, созданный при выполнении лабораторной работы №8, с программой выводящей на экран аргументы командной строки в файл с именем lab09-3.asm и создал исполняемый файл. Загрузил исполняемый файл в отладчик, указав аргументы (рис. ??).



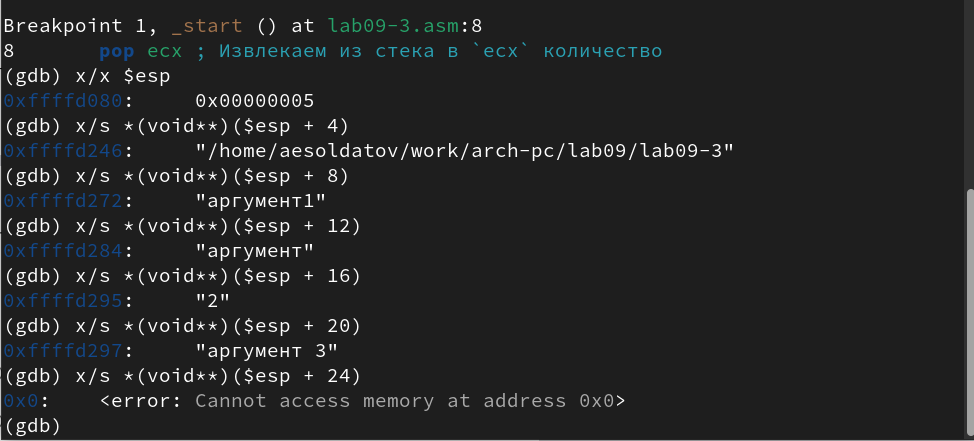
Копирование и запуск файла

Поставил точку останова перед первой инструкцией в программе и запустил ее (рис. ??).



Установка точки останова

Просмотрел адрес вершины стека, который хранится в регистре esp, а также другие позиции стека - по адресу (рис. ??).

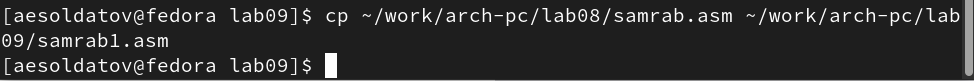


Позиции стека

Количество аргументов командной строки 4, поэтому шаг равен четырем.

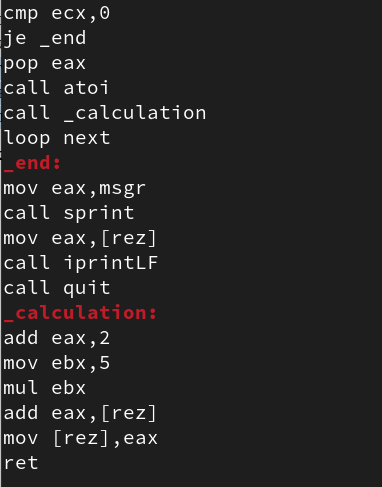
## 4.3 Выполнение заданий для самостоятельной работы

Создал файл “samrab1.asm”, скопировав файл из прошлой лабораторной работы (рис. ??).



Создание файла

Преобразовал программу из лабораторной работы №8, реализовав вычисление значения функции как подпрограмму (рис. ??).



Ввод текста программы

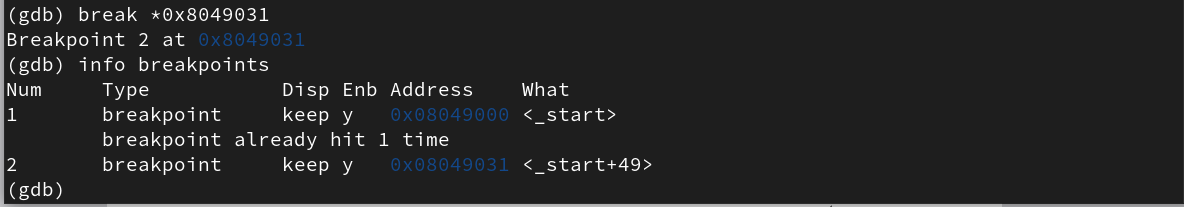
%include 'in\_out.asm'  
SECTION .data  
msgf db "Функция: f(x)=5\*(2+x)",0  
msgr db "Результат: ",0  
rez dd 0d  
SECTION .bss  
rezx resb 80  
SECTION .text  
global \_start  
\_start:  
mov eax,msgf  
call sprintLF  
pop ecx  
pop edx  
sub ecx,1  
next:  
cmp ecx,0  
je \_end  
pop eax  
call atoi  
call \_calculation  
loop next  
\_end:  
mov eax,msgr  
call sprint  
mov eax,[rez]  
call iprintLF  
call quit  
\_calculation:  
add eax,2  
mov ebx,5  
mul ebx  
add eax,[rez]  
mov [rez],eax  
ret

Создал исполняемый файл и проверил его работу (рис. -??).



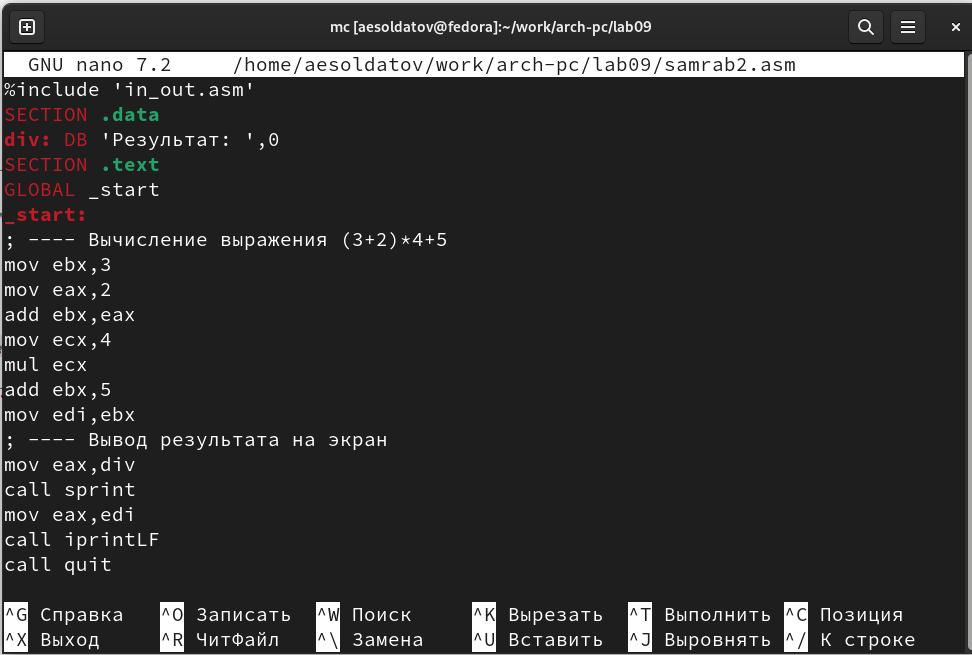
Проверка работы программы

Создал файл “samrab2.asm” (рис. ??).



Создание файла

Ввел текст программы из листинга 9.3 со страницы в ТУИС (рис. ??).



Ввод текста программы

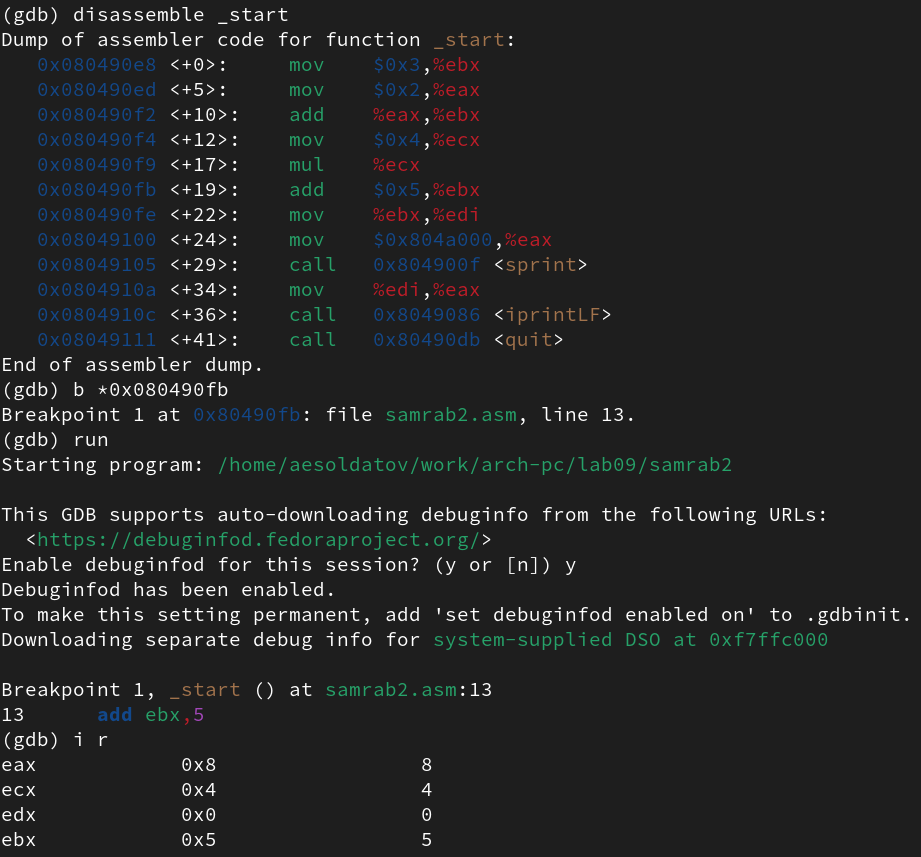
%include 'in\_out.asm'  
SECTION .data  
div: DB 'Результат: ',0  
SECTION .text  
GLOBAL \_start  
\_start:  
; ---- Вычисление выражения (3+2)\*4+5  
mov ebx,3  
mov eax,2  
add ebx,eax  
mov ecx,4  
mul ecx  
add ebx,5  
mov edi,ebx  
; ---- Вывод результата на экран  
mov eax,div  
call sprint  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

Создал исполняемый файл и проверил его работу, программа действительно выдает неверный результат (рис. -??).



Проверка работы программы

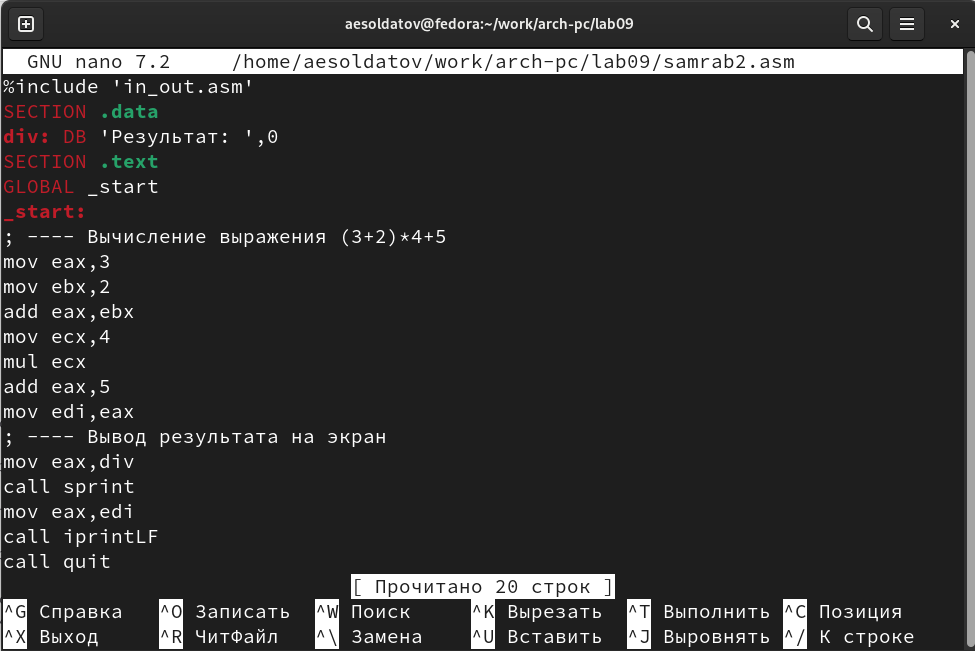
Просмотрел дисассемблированный код программы, поставил точку останова перед прибавлением 5 и открываю значения регистров на данном этапе (рис. -??).



Дисассемблированный код программы

Как можно увидеть, регистр ecx со значением 4 умножается не на ebx, сложенным c eax, а только с eax со значением 2. Значит нужно поменять значения регистров (например присвоить eax значение 3 и просто прибавить 2).

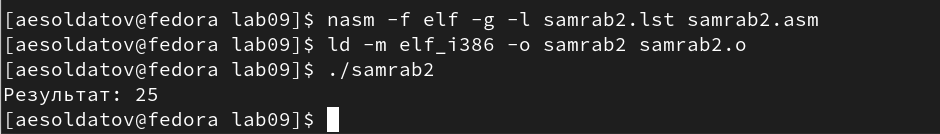
После изменений программа будет выглядеть следующим образом: (рис. -??).



Изменение текста программы

%include 'in\_out.asm'  
SECTION .data  
div: DB 'Результат: ',0  
SECTION .text  
GLOBAL \_start  
\_start:  
; ---- Вычисление выражения (3+2)\*4+5  
mov eax,3  
mov ebx,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

Создал исполняемый файл и проверил его работу, теперь программа работает как надо (рис. -??).



Проверка работы программы

# 5 Выводы

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

# Список литературы

1. GNU Bash Manual [Электронный ресурс]. Free Software Foundation, 2016. URL: <https://www.gnu.org/software/bash/manual/>.

2. Newham C. [Learning the bash Shell: Unix Shell Programming](http://www.amazon.com/Learning-bash-Shell-Programming-Nutshell/dp/0596009658). O’Reilly Media, 2005. 354 с.

3. Zarrelli G. Mastering Bash. Packt Publishing, 2017. 502 с.

4. Robbins A. [Bash Pocket Reference](https://www.ncbi.nlm.nih.gov/pubmed/25246403). O’Reilly Media, 2016. 156 с.

5. Таненбаум Э. Архитектура компьютера. 6-е изд. СПб.: Питер, 2013. 874 с.

6. Таненбаум Э., Бос Х. Современные операционные системы. 4-е изд. СПб.: Питер, 2015. 1120 с.