Отчет по лабораторной работа №10

Группа НПИбд-02-22

Стариков Данила Андреевич

Содержание

# 1 Цель работы

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

# 2 Основная часть

## 2.1 Выполнение лабораторной работы

### 2.1.1 Реализация подпрограмм в NASM

Для выполнения лабораторной работы создали каталог ~/work/study/arch-pc/lab10. В нем создали файл lab10-1.asm и ввели текст из Листинга 1. Также положили в каталог файл in\_out.asm, использованный в лабораторной работе №6.

Листинг 1: . Пример программы с использованием вызова подпрограммы

%include 'in\_out.asm'  
  
SECTION .data  
 msg: DB 'Введите x: ',0  
 result: DB '2x+7=',0  
  
SECTION .bss  
 x: RESB 80  
 rez: 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,[rez]  
 call iprintLF  
   
 call quit  
;------------------------------------------  
; Подпрограмма вычисления  
; выражения "2x+7"  
 \_calcul:  
 mov ebx,2  
 mul ebx  
 add eax,7  
 mov [rez],eax  
 ret ; выход из подпрограммы

Создали исполняемый файл и запустили его (Рис. 1).

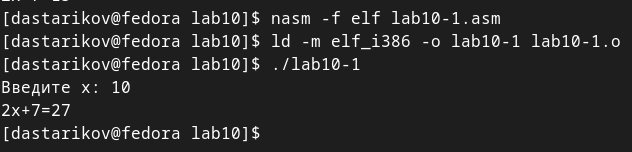


Рис. 1: Результат запуска файла lab10-1.

Затем изменили текст программы, добавив еще одну подпрограмму \_subcalcul, которая вычисляет значение , а в результате вычисляется значение выражения и выводится на экран (Листинг 2).

Листинг 2: Вычисление выражения f(g(x)) с использованием жвух подпрограмм.

%include 'in\_out.asm'  
  
SECTION .data  
 msg: DB 'Введите x: ',0  
 result: DB 'f(x)=2x+7, g(x)= 3x-1, f(g(x)) = ',0  
  
SECTION .bss  
 x: RESB 80  
 rez: 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,[rez]  
 call iprintLF  
 call quit  
;------------------------------------------  
; Подпрограмма вычисления  
; выражения "f(g(x))"  
 \_calcul:  
 call \_subcalcul ; сначала вычисляем g(x)  
 mov ebx,2  
 mul ebx  
 add eax,7  
 mov [rez],eax  
 ret ; выход из подпрограммы  
   
;------------------------------------------  
; Подпрограмма вычисления  
; выражения "g(x)=3x-1"  
 \_subcalcul:  
 mov ebx, 3  
 mul ebx  
 dec eax  
 ret

Создали исполняемый файл и запустили его (Рис. 2).

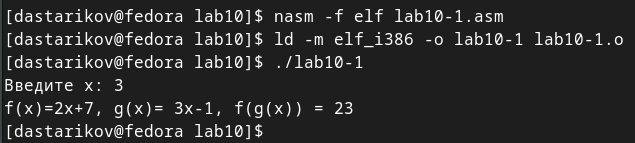


Рис. 2: Вычсиление f(g(x)).

### 2.1.2 Отладка программ с помощью GDB

Создали файл lab10-2.asm и ввели текст из Листинга 3.

Листинг 3: Программа вывода сообщения Hello world!

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

Для работы с отладчиком GDB при трансляции исполняемого файла добавили ключ -g (Рис. 3).

Рис. 3: Компиляция файла lab10-2.asm

Рис. 3: Компиляция файла lab10-2.asm

Загрузили исполняемый файл в отладчик GDB и проверили работы по команде run (Рис. 4).

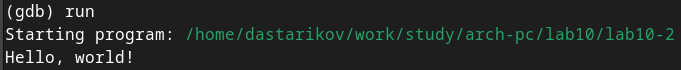


Рис. 4: Обычный запуск программы.

Далее установили брейкпоинт на метку \_start, и еще раз запустили ее (Рис. 5).

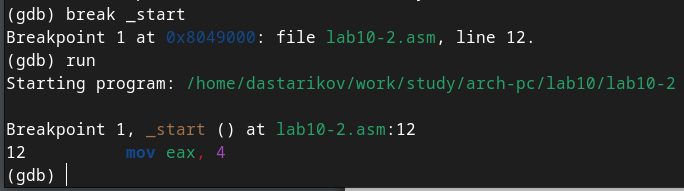


Рис. 5: Установлен брейкпоинт.

Посмотрели дисассемблированный код программы в режиме AT&T и Intel (Рис. 6 и 7). Основные различия синтаксисов: противоположное расположение операнда-источника и операнда-приемника (в Intel - приемник, источник, а в AT&T - источник, приемник); в AT&T регистры пишутся после ‘%’, а непосредственные операнды после ‘$’, в синтаксисе Intel операнды никак не помечаются.



Рис. 6: Дисассемблированный код программы в режиме ATT

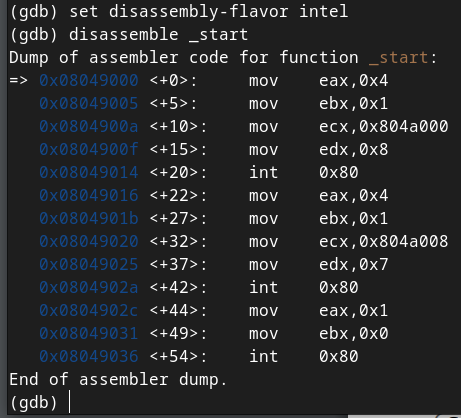


Рис. 7: Дисассемблированный код программы в режиме Intel

Включили режим псевдографики для более удобного анализа (Рис. 8).

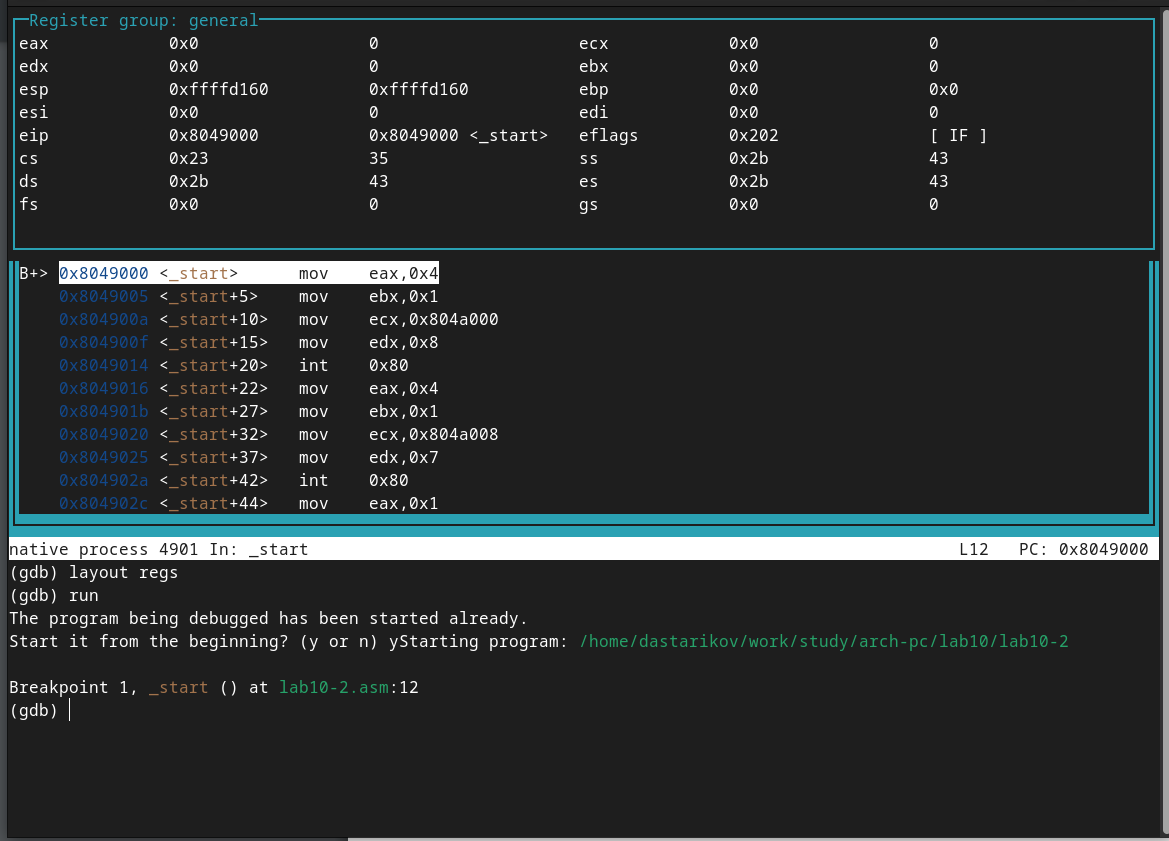


Рис. 8: Режим псевдографики GDB.

Проверили установленные брейкпоинты по команде info breakpoints (Рис. 9).

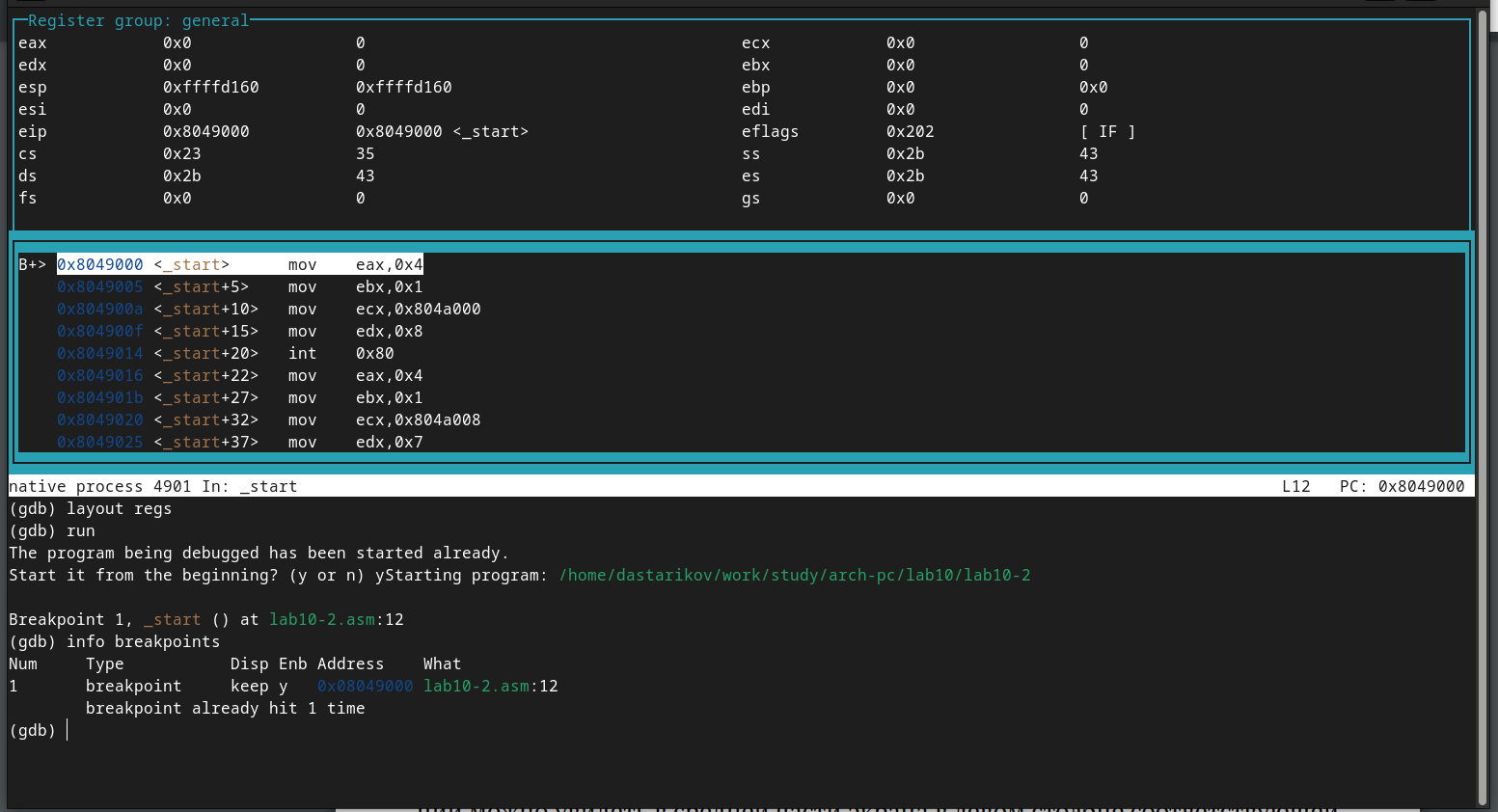


Рис. 9: Список установленных брейкпоинтов.

Добавили еще одну точку остановки по адресу инструкции mov ebx, 0x0 (Рис. 10).

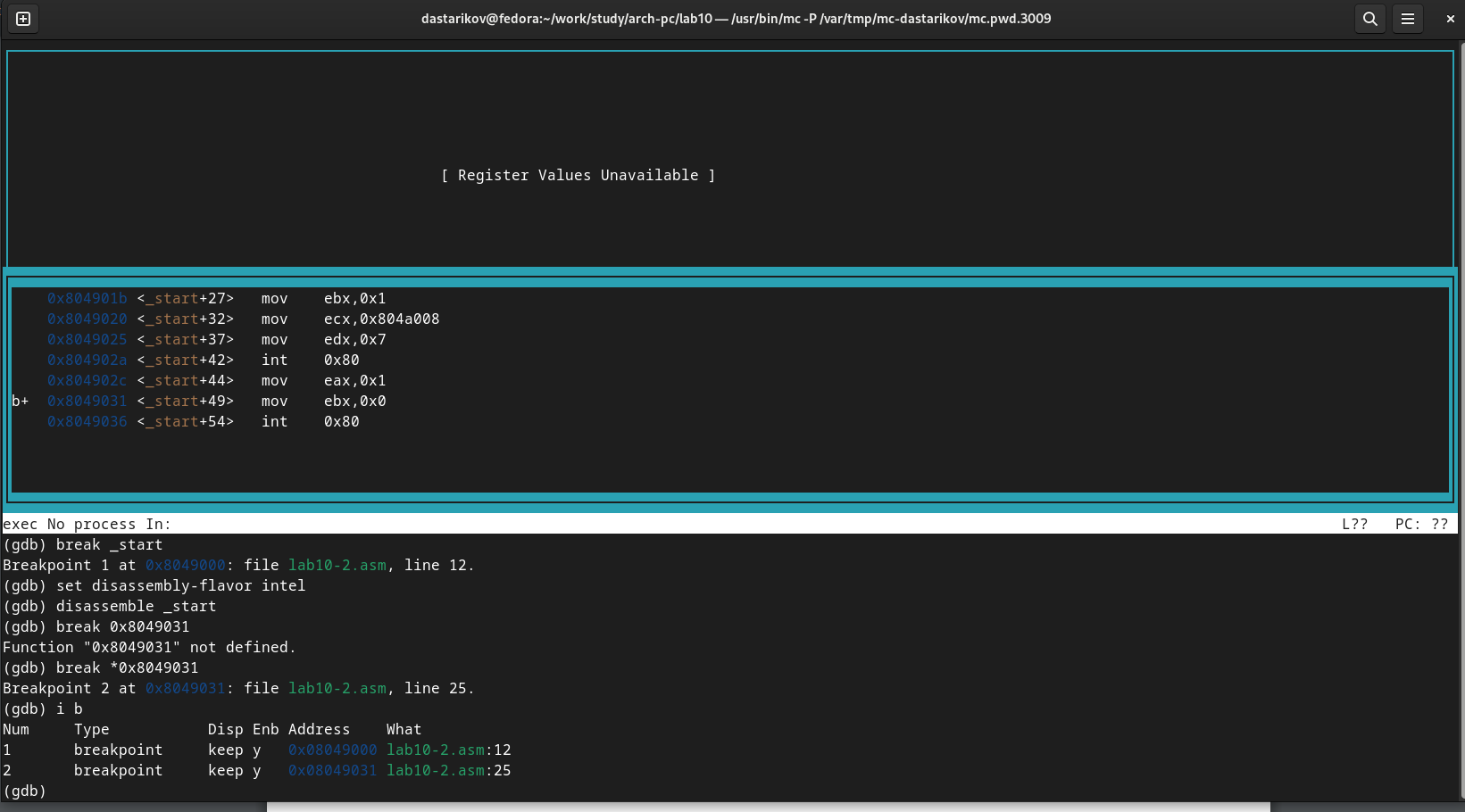


Рис. 10: Добавление новой точки остановки.

С помощью инструкции stepi (сокращенно si) выполнили 5 последующих инструкций. В результате изменяются значения регистров eax, ebx, ecx, edx (Рис. 11).

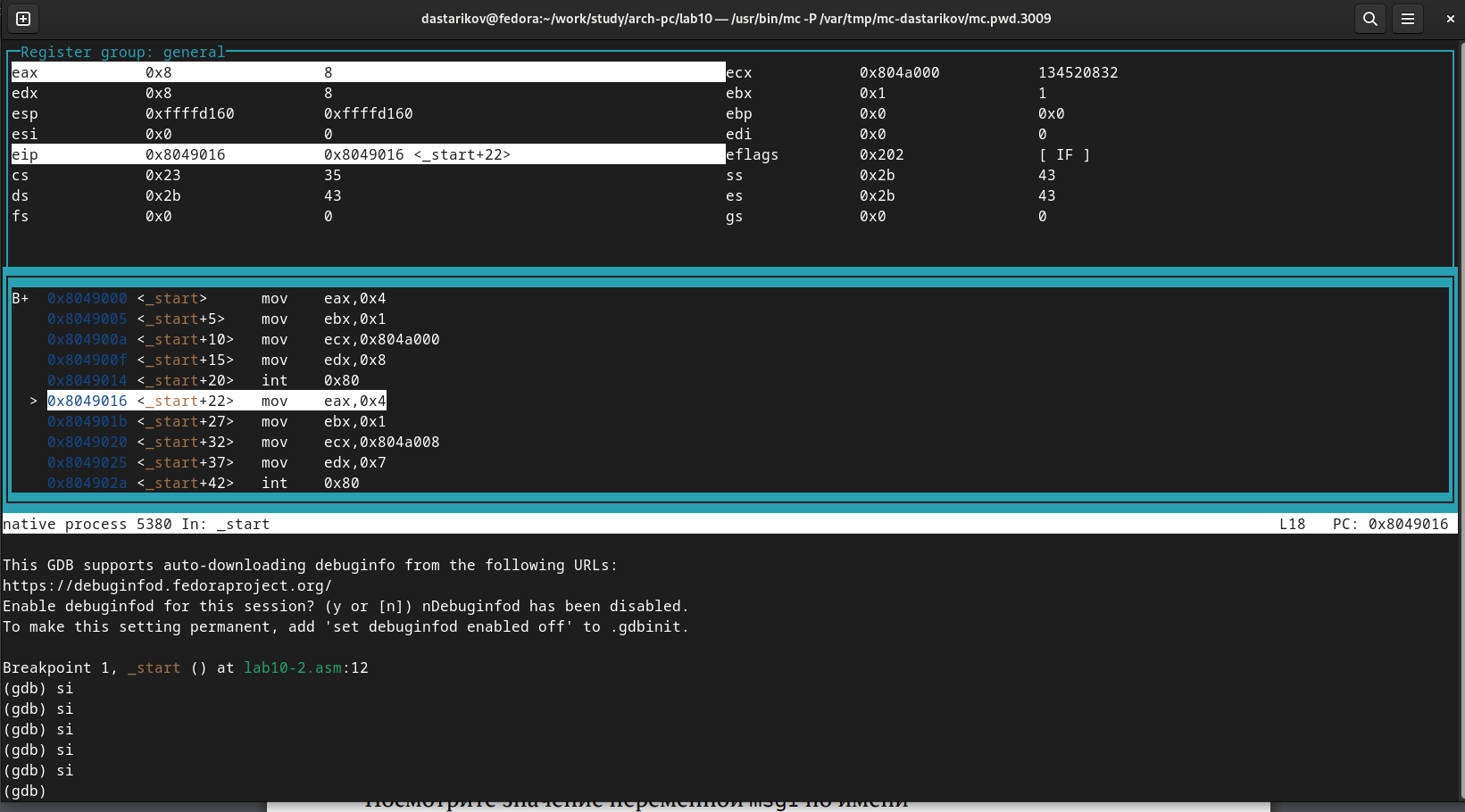


Рис. 11: Выполнение первых 5 инструкций.

Посмотрели значения переменных msg1 и msg2, причем адрес памяти msg1 задали по имени переменной, а адрес памяти msg2 вписали вручную, взяв его из дисассемблированного кода программы (Рис. 12).

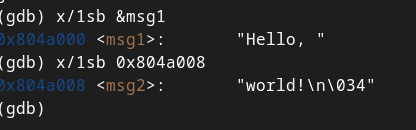


Рис. 12: Вывод значения переменных msg1 и msg2.

С помощью команды set заменили в msg1 первый символ, в переменной msg2 - второй (Рис. 13 и 14).

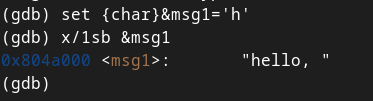


Рис. 13: Изменение значения переменной msg1.

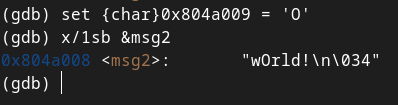


Рис. 14: Изменение значения переменной msg2.

Вывели значения регистра edx в десятичном, шестнадцатиричном и двоичном формате, соответственно (Рис. 15).

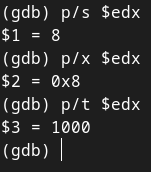


Рис. 15: Значение регистра edx.

Использовав команду set изменили значение регистра ebx сначала на символ ‘2’, а затем на число 2, и сравнили вывод значения регистра в десятичном формате. В результате присвоения регистра значение символа ‘2’, выводится число 50, что соответствует символу в ‘2’ в таблице ASCII (Рис. 16).

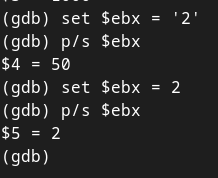


Рис. 16: Изменение значнения регистра ebx.

Затем завершили выполнение программы командой continue (сокращенно c), и вышли из GDB по команде exit.

Для примера обработки аргментов командоной строки в GDB скопировали файл lab9-2.asm из Лабораторной работы №9 (программа выводит на экран аргументы командной строки) в файл lab10-3.asm, создали исполняемый файл и запустили в GDB с ключом --args (Рис. 17).

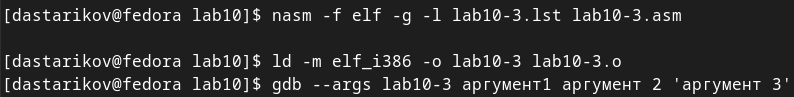


Рис. 17: Загразка в GDB программы lab10-3 с аргументами.

Установили брейкпоинт в начале программы, посмотрели значение регистра esp, равное количеству аргументов командной строки, включая имя программы, и остальные позиции стека (Рис. 18). Каждый элемент стека занимает 4 байта, поэтому для получения следующего элемента стека мы добавляем 4 к адресу вершины.

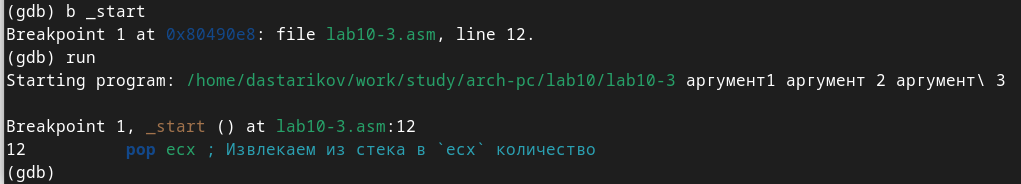


Рис. 18: Значения элементов стека.

## 2.2 Выполнение заданий для самостоятельной работы.

### 2.2.1 Задание 1.

Написали программу lab10-4.asm (Листинг 4), являющуюся измененной версией программы lab09-var12.asm, но вычисление реализовано как подпрограмма (Рис. 19).

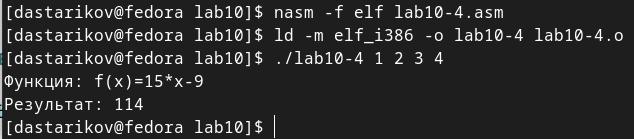


Рис. 19: Результат работы программы lab10-4.

Листинг 4: Программа находит сумму значений функции

%include 'in\_out.asm'  
  
SECTION .data  
 msg1 db "Функция: f(x)=15\*x-9 ",0  
 msg2 db "Результат: ",0  
SECTION .text  
 global \_start  
  
\_start:  
  
 mov eax, msg1  
 call sprintLF  
   
 pop ecx ; Извлекаем из стека в `ecx` количество  
 ; аргументов (первое значение в стеке)  
 pop edx ; Извлекаем из стека в `edx` имя программы  
 ; (второе значение в стеке)  
 sub ecx,1 ; Уменьшаем `ecx` на 1 (количество  
 ; аргументов без названия программы)  
 mov esi, 0 ; Используем `esi` для хранения  
 ; промежуточных сумм  
next:  
 cmp ecx,0h ; проверяем, есть ли еще аргументы  
 jz \_end ; если аргументов нет выходим из цикла  
 ; (переход на метку `\_end`)  
 pop eax ; иначе извлекаем следующий аргумент из стека  
 call atoi ; преобразуем символ в число  
 call \_calcul ; вызываем подпрограмму для вычисления f(x)  
  
 add esi,eax ; добавляем к промежуточной сумме  
 ; след. аргумент `esi=esi+eax`  
 loop next ; переход к обработке следующего аргумента  
\_end:  
 mov eax, msg2 ; вывод сообщения "Результат: "  
 call sprint  
 mov eax, esi ; записываем сумму в регистр `eax`  
 call iprintLF ; печать результата  
 call quit ; завершение программы  
  
\_calcul: ; подпрограмма для вычисления f(x)=15\*x-9  
 mov ebx, 15  
 mul ebx  
 sub eax, 9  
 ret

### 2.2.2 Задание 2.

Создали программу lab10-5.asm по Листингу 5, вычисляющую выражение , но неверно (Рис. 20).

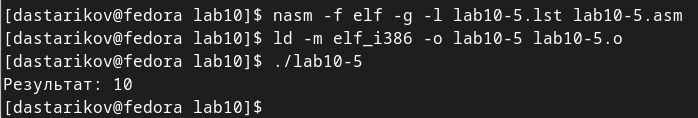


Рис. 20: Результат работы программы lab10-5 (неверный ответ).

Листинг 5: Программа вычисления выражения (3+2)\*4+5 (неверная).

%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

Проанализировали с помощью отладичка GDB изменение значений регистров, чтобы найти ошибку. Установили брейкпоинт b \_start, включили режим псевдографики и начали поочердно выполнять команды и следить за значениями регистров. Первая ошибка - результат записан в регистре ebx (команда add ebx, eax), но команда mul всегда перемножает значение регистра eax с указанным сомножителем, поэтому mul ecx дает неверный результат: . (Рис. 21). Затем на убеждении, что регистр ebx содержит корректный результат вычислений, к нему добавляется число 5, результат запоминается в регистре edi, а затем выводится (Рис. 22).

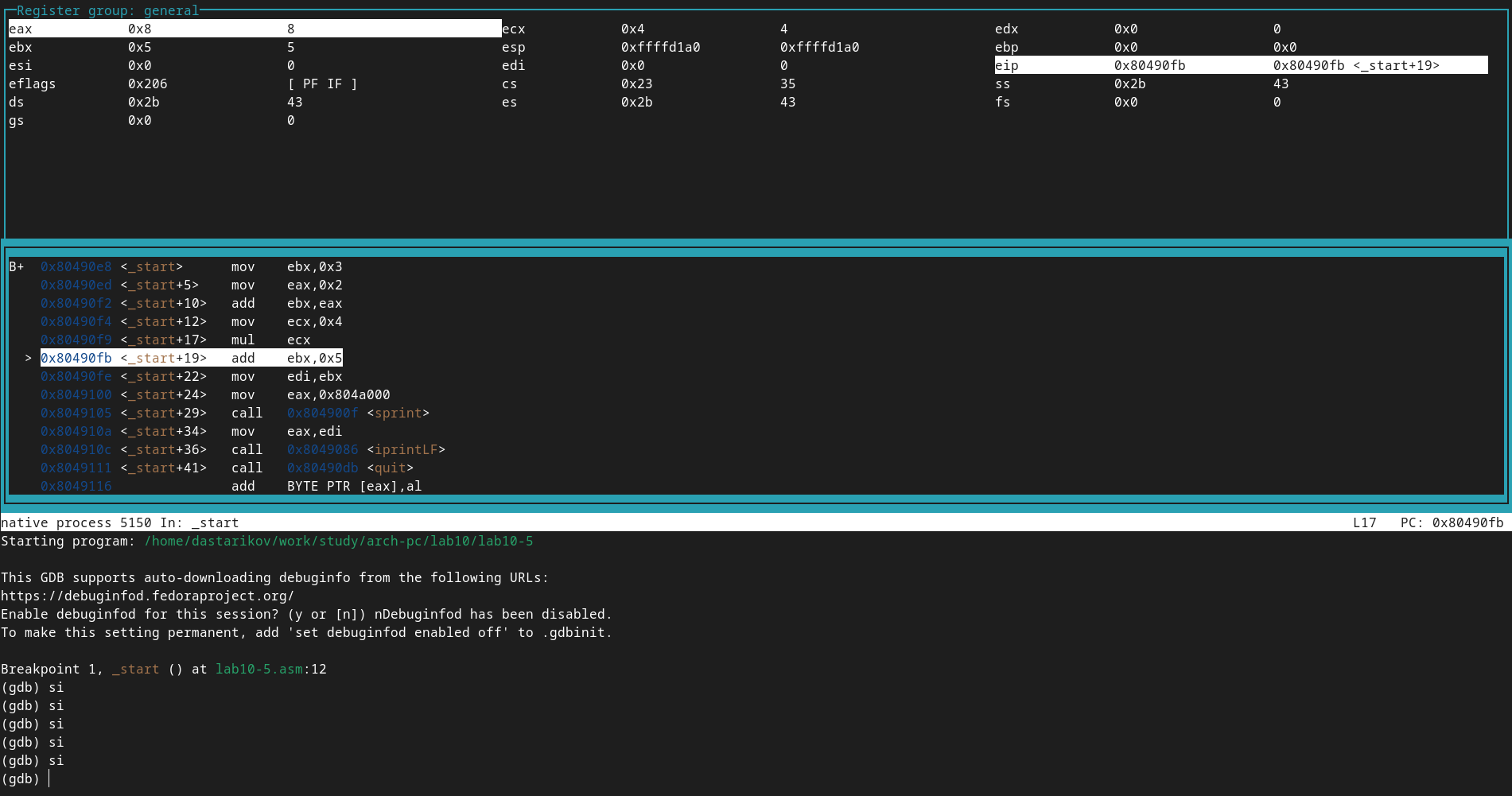


Рис. 21: Некорректное использование команды mul.

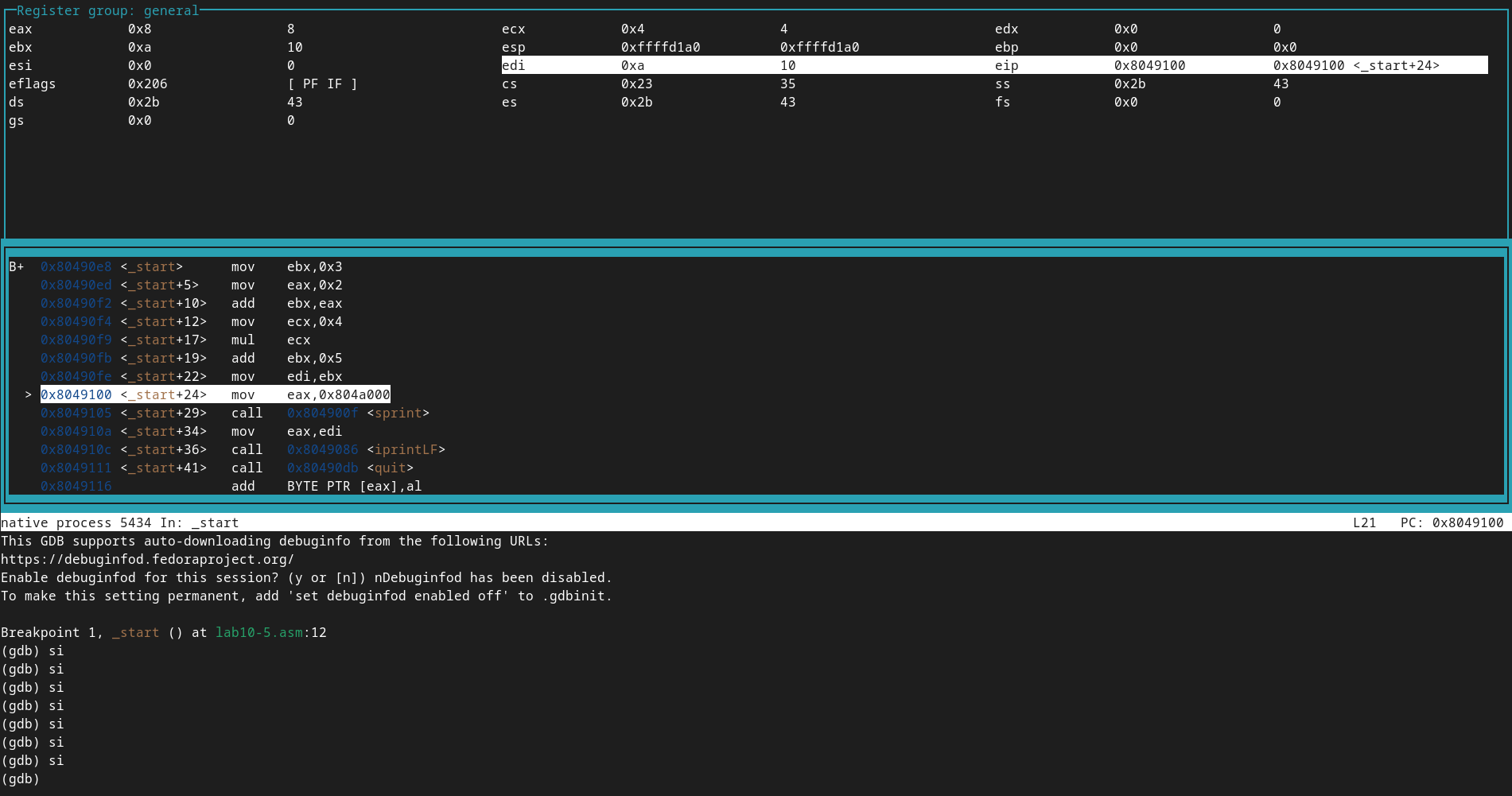


Рис. 22: В регистре ebx содержится неверный ответ.

Исправили текст программы (Листинг [-lst:lst06]) и убедились в верном вычислении резульата (Рис [-fig:fig24]).

Листинг 6: Программа вычисления выражения (3+2)\*4+5 (неверная).

%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

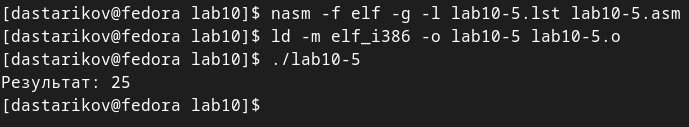


Рис. 23: Правильный результат вычисления выражения.

Созданные файлы \*.asm скопировали в каталог ~/work/study/2022-2023/“Архитектура компьютера”/archpc/labs/lab10/ и загрузили на Github.

# 3 Выводы

В рамках лабораторной работы получили практические навыки использования подпрограмм и познакомились с методами отладки для более глубокого анализа работы программы.