Отчёт по лабораторной работе №10

Дисциплина: Архитектура компьютера

Мишина Анастасия Алексеевна

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

Для начала создадим каталог для программ 10-ой лабораторной работы, перейдем в него и создадим файл lab10-1.asm (рис. 1).

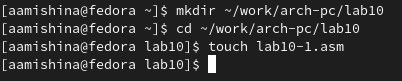


Рис. 1: Создание каталога и файла

Вводим текст программы из листинга 10.1 в наш файл. Создадим и запустим исполняемый файл, удостоверимся в его работе (рис. 2).

Программа lab10-1.asm:

%include 'in\_out.asm'  
SECTION .data  
msg: DB 'Введите x: ',0  
result: DB '2x+7=',0  
SECTION .bss  
x: RESB 80  
rezs: 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 [rez],eax  
ret ; выход из подпрограммы

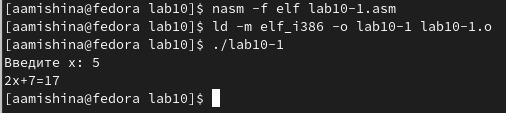


Рис. 2: Работа с файлом lab10-1.asm

Меняем текст программы, добавляя подпрограмму \_subcalcul для вычисления выражения f(g(x)). Создаем исполняемый файл, программа отрабатывает успешно (рис. 3).

Измененная программа lab10-1.asm:

%include 'in\_out.asm'  
SECTION .data  
msg: DB 'Введите x: ',0  
result: DB '2(3x-1)+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:  
call \_subcalcul  
mov ebx,2  
mul ebx  
add eax,7  
mov [rez],eax  
ret ; выход из подпрограммы  
  
\_subcalcul:  
mov ebx, 3  
mul ebx  
sub eax, 1  
ret

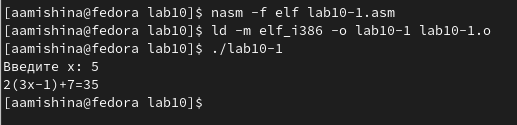


Рис. 3: Работа с измененным файлом lab10-1.asm

Теперь создадим файл lab10-2.asm, вводим в него текст листинга 10.2. Создаем исполняемый файл с ключом ‘-g’ для работы с GDB. Загружаем исполняемый файл в отладчик gdb (рис. 4). Запускаем программу с помощью команды run, она отрабатывает успешно (рис. 5).

Программа lab10-2.asm:

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

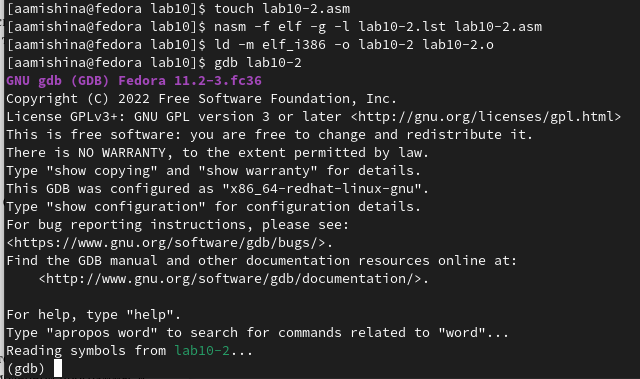


Рис. 4: Работа с файлом lab10-2.asm

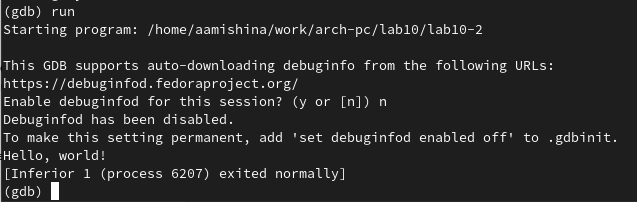


Рис. 5: Запуск программы lab10-2.asm

Устанавливаем брейкпоинт на метку \_start и запускаем программу еще раз. Посмотрим на дисассимилированный код программы с помощью команды disassemble начиная с метки \_start (рис. 6). Переключимся на изображение с Intel’овским синтаксисом (рис. 7). Видим отличие в третьем столбце в изображении названий регистров и переменных. Также отличается порядок переменная-регистр и наоборот регистр-переменная.

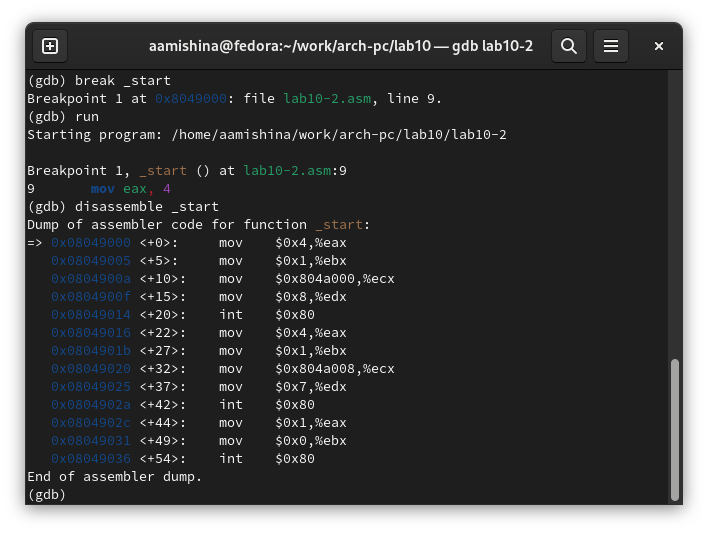


Рис. 6: Брейкпоинт и дисассимилированный код программы

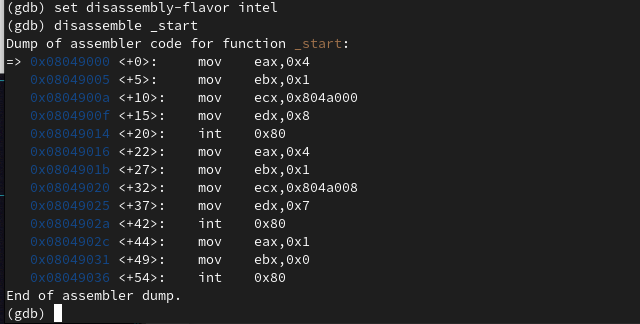


Рис. 7: Intel’овский синтаксис

Далее включаем режим псевдографики для более удобного анализа программы. С помощью команды info breakpoints смотрим, что уже установили одну точку остновки (рис. 8). Устанавливаем еще одну, используя адрес инструкции. Снова смотрим информацию о всех брейкпоинтах (рис. 9).

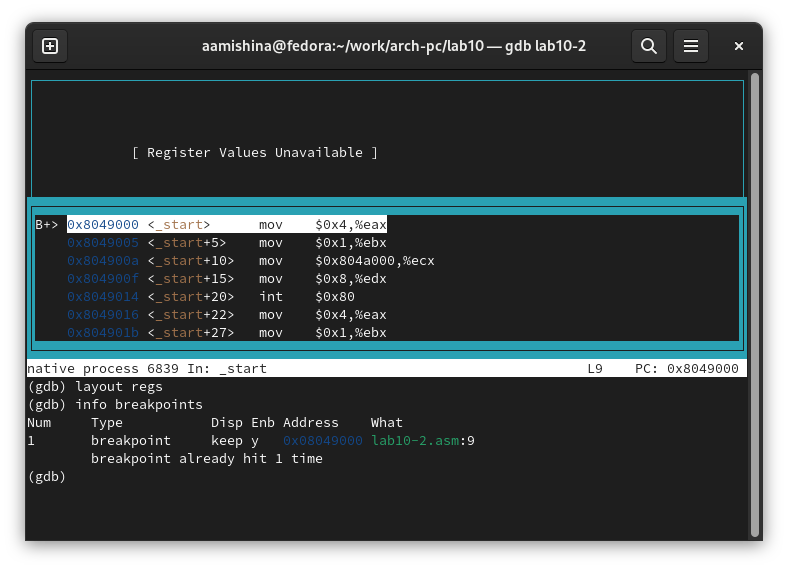


Рис. 8: Информация о брейкпоинте

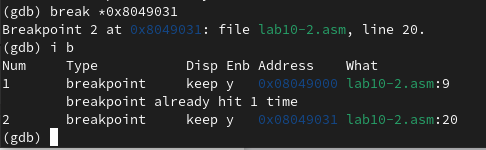


Рис. 9: Установка второго брейкпоинта, вывод информации

Далее выполняем ровно 5 раз команду si, видим изменения в регистрах eax, ebx, ecx, edx, esp (рис. 10). Смотрим содержимое регистров (рис. 11).

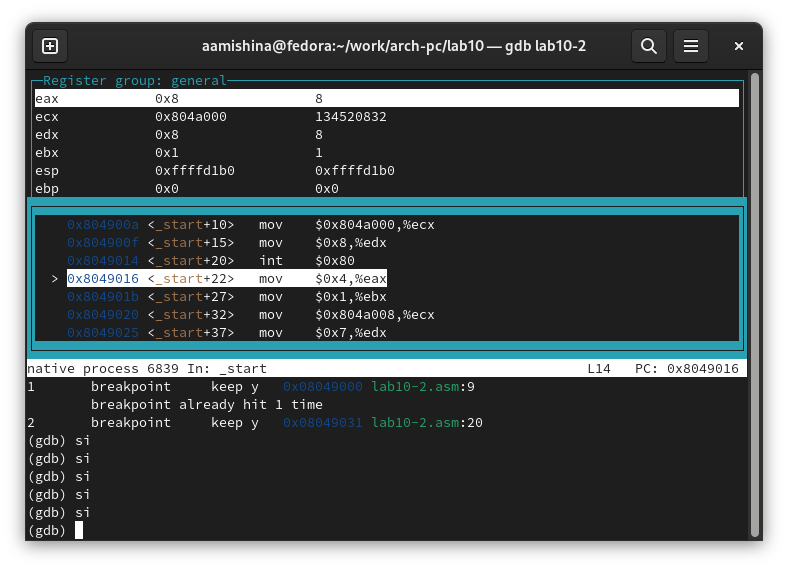


Рис. 10: Выполнение команды si

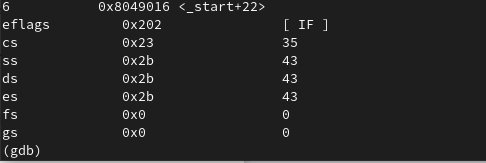


Рис. 11: Содержимое регистров

Теперь посмотрим значения переменных msg1 по имени и msg2 по адресу. Также смотрим инструкцию mov ecx, msg2 (рис. 12).

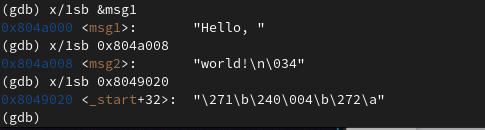


Рис. 12: Содержимое переменных и инструкции

Меняем символы в перменных msg1 и msg2 (рис. 13).

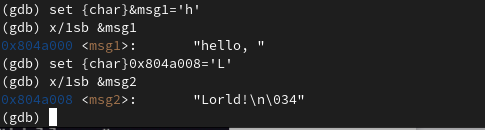


Рис. 13: Замена символов в перменных msg1 и msg2

Смотрим значения нескольких регистров (рис. 14). Самостоятельно смотрим значение регистра edx в шестнадцатеричном формате, в двоичном формате и в символьном виде (рис. 15).

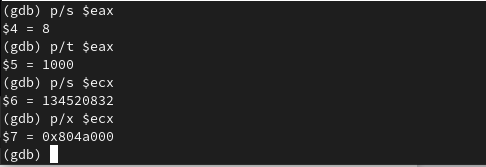


Рис. 14: Просмотр значений нескольких регистров

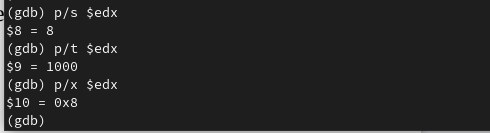


Рис. 15: Просмотр значений регистра edx

С помощью команды set меняем значение регистра ebx. Сначала выводится 50, так как это ASCII кодировка символа “2”, а затем 2 - уже как число (рис. 16).

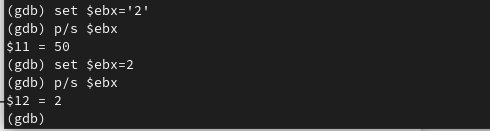


Рис. 16: Замена значения регистра ebx

Завершаем выполнение программы (рис. 17).

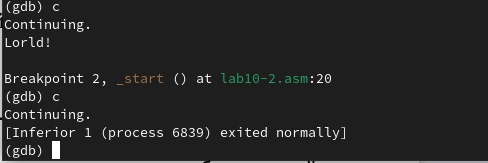


Рис. 17: Завершение работы программы

Копируем файл lab9-2.asm в файл с именем lab10-3.asm. Создаем исполняемый файл с отладчиком gdb. Загружаем исполняемый файл в отладчик, указав аргументы (рис. 18).

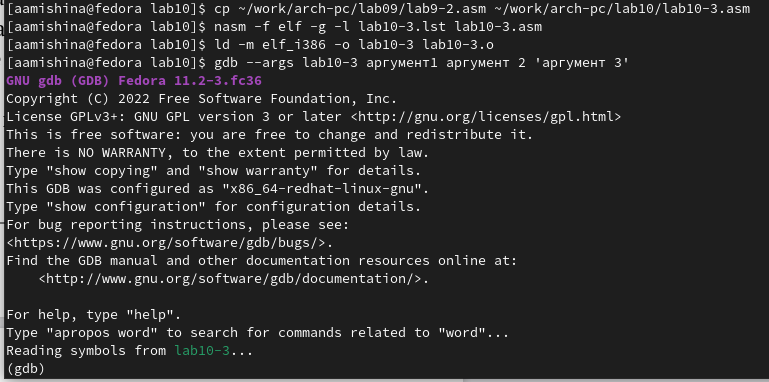


Рис. 18: Работа с файлом lab10-3.asm

Устанавливаем точку останова перед первой инструкцией в программе и запускаем ее. Смотрим, что хранится в регистре esp, получаем 5 аргументов (рис. 19).

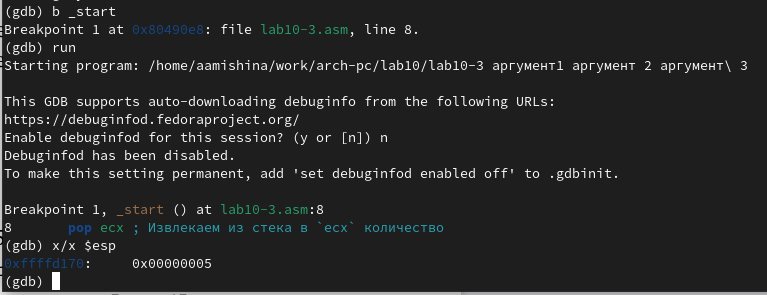


Рис. 19: Просмотр регистра esp

Теперь посмотрим остальные позиции стека (рис. 20). Шаг изменения равен 4, потому что шаг равен размеру переменной (4 байта).

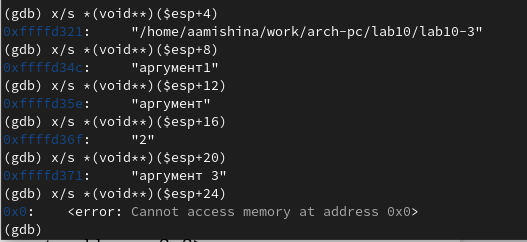


Рис. 20: Просмотр остальных позиций стека

# 3 Выполнение заданий самостоятельной работы

Для начала дорабатываем программу из 9-ой лабораторной работы. Добавляем подпрограмму для вычисления значения функции (рис. 21).

Программа lab10my.asm:

%include 'in\_out.asm'  
SECTION .data  
msg db "Результат: ", 0  
func: db 'f(x) = 12x - 7', 0  
  
SECTION .text  
global \_start  
\_start:  
mov eax, func  
call sprintLF  
  
pop ecx  
pop edx  
sub ecx, 1  
mov esi, 0  
  
next:  
cmp ecx, 0h  
jz \_end  
  
pop eax  
call atoi  
  
call \_calc  
add esi, eax  
  
loop next  
  
\_end:  
mov eax, msg  
call sprint  
  
mov eax, esi  
call iprintLF  
  
call quit  
  
\_calc:  
mov ebx, 12  
mul ebx  
  
sub eax, 7  
ret

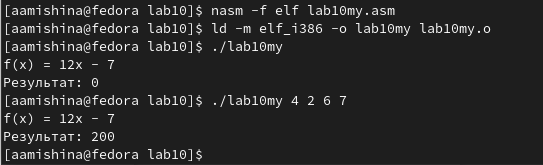


Рис. 21: Тестирование программы lab10my.asm

Вторым заданием было определить ошибку в программе, используя отладчик GDB и анализируя изменение значений регистров. Стало понятно, что в нескольких местах следует поменять регистр ebx на eax. Измененная программа отработала успешно (рис. 22), (рис. 23), (рис. 24).

Нерабочий код 10.3:

%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

Исправленный мной код:

%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

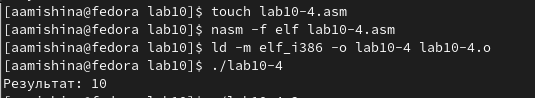


Рис. 22: Убеждаемся в неработоспособности программы

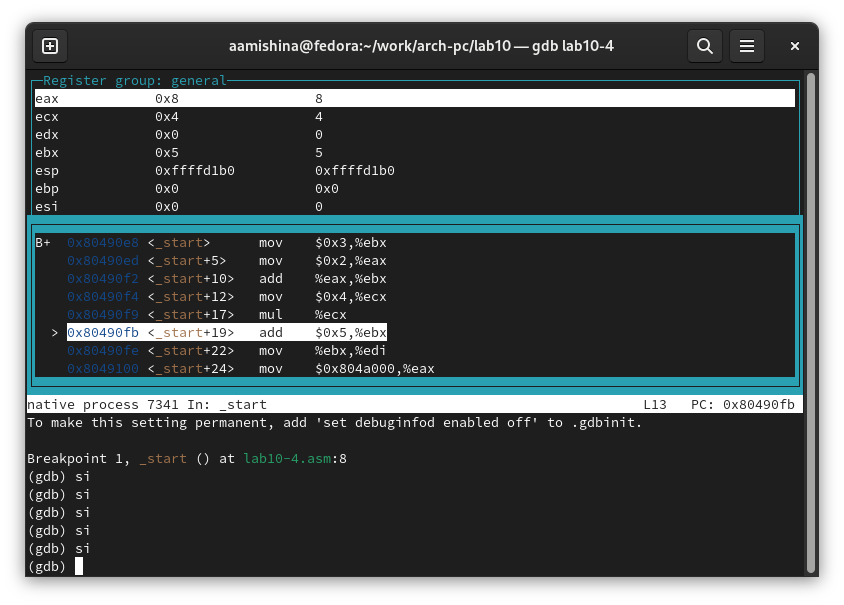


Рис. 23: Поиск ошибки

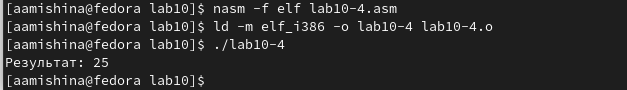


Рис. 24: Успешная отработка программы

# 4 Выводы

В ходе выполнения данной лабораторной работы я приобрела навыки написания программ с использованием подпрограмм. Также я познакомилась с методами отладки при помощи GDB и его основными возможностями. Вся моя работа была записана и прокомментирована мной в данной лабораторной.