

Лабораторная работа №9

Понятие подпрограммы. Отладчик GDB.

Жукова Арина Александровна

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
2.1	Реализация подпрограмм в NASM	6
2.2	Отладка программ с помощью GDB	7
2.3	Добавление точек останова	10
2.4	Работа с данными программы в GDB	11
2.5	Обработка аргументов командной строки в GDB	13
3	Задания для самостоятельной работы	15
4	Выводы	18
	Список литературы	19

Список иллюстраций

2.1	Проверка работы программы	6
2.2	Проверка работы программы	7
2.3	Получение исполняемого файла	7
2.4	Загрузка файла в отладчик gbd	7
2.5	Проверка работы программы	8
2.6	Запуск программ с брейкпоинтом	8
2.7	Просмотр дисассимилированного кода	8
2.8	Отображение команд с Intel’овским синтаксисом	9
2.9	Окна в режиме псевдографики	10
2.10	Работа команды info breakpoints	10
2.11	Информация о всех точках останова	10
2.12	Содержимое регистров	11
2.13	Значение переменной по имени	11
2.14	Значение переменной по адресу	11
2.15	Замена значения переменной	11
2.16	Замена символа	12
2.17	Значение регистра edx	12
2.18	Изменение значения	12
2.19	Изменение значения	13
2.20	Создание исполняемого файла	13
2.21	Загрузка исполняемого файла в отладчик	13
2.22	Установка точки останова	14
2.23	Позиции стека	14
3.1	Проверка нахождения файла	15
3.2	Работа программы	16
3.3	Создание файла, проверка его работы	17
3.4	Работа программы	17

Список таблиц

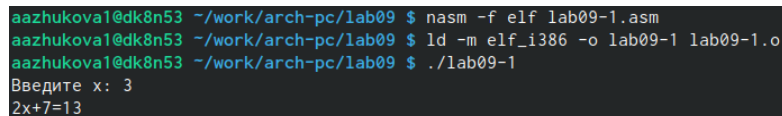
1 Цель работы

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 исполняемый файл и проверяем его работу (рис. 2.1).



```
aazhukova1@dk8n53 ~/work/arch-pc/lab09 $ nasm -f elf lab09-1.asm
aazhukova1@dk8n53 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-1 lab09-1.o
aazhukova1@dk8n53 ~/work/arch-pc/lab09 $ ./lab09-1
Введите x: 3
2x+7=13
```

Рис. 2.1: 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 подпрограммы
;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. 2.2).

```
aazhukova1@dk8n53 ~/work/arch-pc/lab09 $ nasm -f elf lab09-1.asm
aazhukova1@dk8n53 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-1 lab09-1.o
aazhukova1@dk8n53 ~/work/arch-pc/lab09 $ ./lab09-1
Введите x: 3
f(x)=2x+7, g(x)=3x-1, f(g(x))=23
```

Рис. 2.2: Проверка работы программы

2.2 Отладка программ с помощью GDB

Создаём файл lab09-2.asm с текстом программы из Листинга 9.2. Получаем исполняемый файл, для работы с GDB добавляем отладочную информацию при помощи ключа ‘-g’ (рис. 2.3).

```
aazhukova1@dk8n53 ~/work/arch-pc/lab09 $ nasm -f elf -g -l lab09-2.lst lab09-2.asm
aazhukova1@dk8n53 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-2 lab09-2.o
```

Рис. 2.3: Получение исполняемого файла

Загружаем исполняемый файл в отладчик gdb, проверяем работу программы в оболочке GDB при помощи команды run (рис. 2.4-2.5).

```
aazhukova1@dk8n53 ~/work/arch-pc/lab09 $ gdb lab09-2
GNU gdb (Gentoo 12.1 vanilla) 12.1
Copyright (C) 2022 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
(gdb)
```

Рис. 2.4: Загрузка файла в отладчик gdb

```
(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/a/a/aazhukova1/work/arch-pc/lab09/lab09-2
Hello, world!
[Inferior 1 (process 4880) exited normally]
```

Рис. 2.5: Проверка работы программы

Устанавливаем брейкпоинт на метку `_start`, для более подробного анализа программы, и запускаем файл (рис. 2.6).

```
(gdb) break _start
Breakpoint 1 at 0x08049000: file lab09-2.asm, line 12.
(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/a/a/aazhukova1/work/arch-pc/lab09/lab09-2

Breakpoint 1, _start () at lab09-2.asm:12
12      mov     eax, 4
```

Рис. 2.6: Запуск программ с брейкпоинтом

Просматриваем дисассимилированный код программы с помощью команды `disassemble` начиная с метки `_start` (рис. 2.7).

```
(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x08049000 <+0>:      mov     $0x4,%eax
0x08049005 <+5>:      mov     $0x1,%ebx
0x0804900a <+10>:     mov     $0x804a000,%ecx
0x0804900f <+15>:     mov     $0x8,%edx
0x08049014 <+20>:     int     $0x80
0x08049016 <+22>:     mov     $0x4,%eax
0x0804901b <+27>:     mov     $0x1,%ebx
0x08049020 <+32>:     mov     $0x804a008,%ecx
0x08049025 <+37>:     mov     $0x7,%edx
0x0804902a <+42>:     int     $0x80
0x0804902c <+44>:     mov     $0x1,%eax
0x08049031 <+49>:     mov     $0x0,%ebx
0x08049036 <+54>:     int     $0x80
End of assembler dump.
```

Рис. 2.7: Просмотр дисассимилированного кода

Переключаемся на отображение команд с Intel'овским синтаксисом (рис. 2.8).


```

(gdb) set disassembly-flavor intel
(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x08049000 <+0>:      mov     eax,0x4
    0x08049005 <+5>:      mov     ebx,0x1
    0x0804900a <+10>:     mov     ecx,0x804a000
    0x0804900f <+15>:     mov     edx,0x8
    0x08049014 <+20>:     int     0x80
    0x08049016 <+22>:     mov     eax,0x4
    0x0804901b <+27>:     mov     ebx,0x1
    0x08049020 <+32>:     mov     ecx,0x804a008
    0x08049025 <+37>:     mov     edx,0x7
    0x0804902a <+42>:     int     0x80
    0x0804902c <+44>:     mov     eax,0x1
    0x08049031 <+49>:     mov     ebx,0x0
    0x08049036 <+54>:     int     0x80
End of assembler dump.

```

Рис. 2.8: Отображение команд с Intel'овским синтаксисом

Intel'овский синтаксис отличается от синтаксиса АТТ:

- 1) порядком операндов, в АТТ 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. 2.9).

```

--Register group: general--
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffc330 0xffffc330

B+> 0x8049000 <_start> mov $0x4,%eax
0x8049005 <_start+5> mov $0x1,%ebx
0x804900a <_start+10> mov $0x804a000,%ecx
0x804900f <_start+15> mov $0x8,%edx
0x8049014 <_start+20> int $0x80

```

Рис. 2.9: Окна в режиме псевдографики

2.3 Добавление точек останова

Проверим установку точки останова с помощью команды `info breakpoints` (рис. 2.10).

```

(gdb) info breakpoints
Num      Type           Disp Enb Address      What
1        breakpoint     keep y   0x08049000 lab09-2.asm:12
breakpoint already hit 1 time

```

Рис. 2.10: Работа команды `info breakpoints`

Установим ещё одну точку останова по адресу инструкции, посмотрим информацию по всем установленным точкам (рис. 2.11).

```

breakpoint already hit 1 time
(gdb) break *0x8049031
Breakpoint 2 at 0x8049031: file lab09-2.asm, line 25.
(gdb) i b
Num      Type           Disp Enb Address      What
1        breakpoint     keep y   0x08049000 lab09-2.asm:12
breakpoint already hit 1 time
2        breakpoint     keep y   0x08049031 lab09-2.asm:25

```

Рис. 2.11: Информация о всех точках останова

2.4 Работа с данными программы в GDB

Посмотрим содержимое регистров при помощи команды `info registers` (рис. 2.12).

```
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffc330 0xffffc330
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8049000 0x8049000 <_start>
eflags   0x202     [ IF ]
cs       0x23     35
ss       0x2b     43
ds       0x2b     43
es       0x2b     43
--Type <RET> for more, q to quit, c to continue without paging--
```

Рис. 2.12: Содержимое регистров

Посмотрим значение переменной `msg1` по имени (рис. 2.13).

```
(gdb) x/1sb &msg1
0x804a000 <msg1>: "Hello, "
```

Рис. 2.13: Значение переменной по имени

Посмотрим значение переменной `msg2` по адресу (рис. 2.14).

```
(gdb) x/1sb 0x804a008
0x804a008 <msg2>: "world!\n\034"
```

Рис. 2.14: Значение переменной по адресу

Изменим значение переменной `msg1` при помощи команды `set` (рис. 2.15).

```
(gdb) set {char}0x804a000='h'
(gdb) x/1sb &msg1
0x804a000 <msg1>: "hello, "
```

Рис. 2.15: Замена значения переменной

Заменяем первый символ во второй переменной msg2 на К (рис. 2.16).

```
(gdb) set {char}0x804a008='K'  
(gdb) x/1sb &msg2  
0x804a008 <msg2>: "Kor1d!\n\034"
```

Рис. 2.16: Замена символа

Выведем в различных форматах (в шестнадцатеричном формате, в двоичном формате и в символьном виде) значение регистра edx (рис. 2.17).

```
(gdb) p/x $edx  
$2 = 0x0  
(gdb) p/t $edx  
$3 = 0  
(gdb) p/s $edx  
$4 = 0
```

Рис. 2.17: Значение регистра edx

Изменим значение регистра ebx при помощи команды set (рис. 2.18).

```
(gdb) set $ebx='2'  
(gdb) p/s $ebx  
$6 = 50  
(gdb) set $ebx=2  
(gdb) p/s $ebx  
$7 = 2
```

Рис. 2.18: Изменение значения

В первом случае мы меняем значение регистра на символ '2', и нам выводится значение 50 в соответствии с таблицей ASCII, а во втором значении 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. 2.19).

```
aazhukova1@dk8n53 ~/work/arch-pc/lab09 $ ls
in_out.asm  lab09-1.asm  lab09-2    lab09-2.lst  lab09-3    lab09-3.lst
lab09-1     lab09-1.o   lab09-2.asm lab09-2.o   lab09-3.asm lab09-3.o
```

Рис. 2.19: Изменение значения

Создадим исполняемый файл (рис. ??).

```
aazhukova1@dk8n53 ~/work/arch-pc/lab09 $ cp ~/work/arch-pc/lab08/lab8-2.asm ~/work/arch-pc/
/lab09/lab09-3.asm
aazhukova1@dk8n53 ~/work/arch-pc/lab09 $ nasm -f elf -g -l lab09-3.lst lab09-3.asm
aazhukova1@dk8n53 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-3 lab09-3.o
```

Рис. 2.20: Создание исполняемого файла

Загрузим исполняемый файл в отладчик, указав аргументы, при помощи ключа `--args` (рис. 2.21).

```
aazhukova1@dk8n53 ~/work/arch-pc/lab09 $ gdb --args lab09-3 аргумент1 аргумент 2 'аргумент 3'
GNU gdb (Gentoo 12.1 vanilla) 12.1
Copyright (C) 2022 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
```

Рис. 2.21: Загрузка исполняемого файла в отладчик

Установим точку останова перед первой инструкцией в программе и запустим ее (рис. 2.22).

```

(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab09-3.asm, line 7.
(gdb)

```

Рис. 2.22: Установка точки останова

Посмотрим позиции стека (рис. 3.1).

```

(gdb) x/x $esp
0xffffc2e0: 0x00000005
(gdb) x/s *(void**)(esp + 4)
0xffffc57e: "/afs/.dk.sci.pfu.edu.ru/home/a/aazhukova1/work/arch-pc/lab09/lab09-3"
(gdb) x/s *(void**)(esp + 8)
0xffffc5c5: "аргумент1"
(gdb) x/s *(void**)(esp + 12)
0xffffc5d7: "аргумент"
(gdb) x/s *(void**)(esp + 16)
0xffffc5e8: "2"
(gdb) x/s *(void**)(esp + 20)
0xffffc5ea: "аргумент 3"
(gdb) x/s *(void**)(esp + 24)
0x0: <error: Cannot access memory at address 0x0>
(gdb)

```

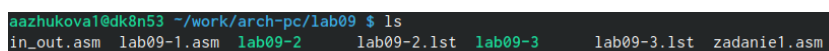
Рис. 2.23: Позиции стека

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 $\text{f}(x)$ как подпрограмму.

Копируем файл задания №1 для самостоятельной работы в папку с заданиями лабораторной работы под именем `zadanie1.asm` (рис. 3.2).



```
aazhukova1@dk8n53 ~/work/arch-pc/lab09 $ ls
in_out.asm  lab09-1.asm  lab09-2      lab09-2.lst  lab09-3      lab09-3.lst  zadanie1.asm
```

Рис. 3.1: 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. 3.2).

```

aazhukova1@dk8n53 ~/work/arch-pc/lab09 $ nasm -f elf zadanie1.asm
aazhukova1@dk8n53 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o zadanie1 zadanie1.o
aazhukova1@dk8n53 ~/work/arch-pc/lab09 $ ./zadanie1 2
Результат: 19
aazhukova1@dk8n53 ~/work/arch-pc/lab09 $ ./zadanie1 1 2
Результат: 36
aazhukova1@dk8n53 ~/work/arch-pc/lab09 $ ./zadanie1 1 2 3
Результат: 57

```

Рис. 3.2: Работа программы

2. В листинге 9.3 приведена программа вычисления выражения $(3 + 2) \times 4 + 5$. 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 отладочную информацию, проверяем корректность работы программы. Выводиться неверный ответ (Результат: 10), верный (Результат: 29) (рис. 3.3).


```

aazhukova1@dk8n53 ~/work/arch-pc/lab09 $ touch zadanie2.asm
aazhukova1@dk8n53 ~/work/arch-pc/lab09 $ nasm -f elf zadanie2.asm
aazhukova1@dk8n53 ~/work/arch-pc/lab09 $ nasm -f elf -g -l zadanie2.lst zadanie2.asm
aazhukova1@dk8n53 ~/work/arch-pc/lab09 $ d -m elf_i386 -o zadanie2 zadanie2.o
bash: d: команда не найдена
aazhukova1@dk8n53 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o zadanie2 zadanie2.o
aazhukova1@dk8n53 ~/work/arch-pc/lab09 $ ./zadanie2
Результат: 10

```

Рис. 3.3: Создание файла, проверка его работы

Исправим ошибку в программе 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. 3.4)

```

aazhukova1@dk8n53 ~/work/arch-pc/lab09 $ ./zadanie2
Результат: 25

```

Рис. 3.4: Работа программы

4 Выводы

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

Список литературы