

Лабораторная работа №9

Архитектура компьютера

Иванов Александр Олегович

Содержание

1	Цель работы	5
2	Теоретическое введение	6
3	Выполнение лабораторной работы	8
4	Самостоятельная работа	13
5	Выводы	15

Список иллюстраций

3.1	Окно MidnightCommander	8
3.2	запуск lab09-1	9
3.3	запуск lab09-2	9
3.4	запуск lab09-2	9
3.5	окно gdb	9
3.6	окно gdb	10
3.7	окно gdb	10
3.8	окно gdb	10
3.9	окно gdb	11
3.10	окно gdb	11
3.11	окно gdb	11
3.12	окно gdb	11
3.13	окно gdb	12
3.14	окно gdb	12
4.1	окно Midnightcommander	13
4.2	запуск lab08-4	14

Список таблиц

1 Цель работы

Приобрести навыки 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 четыре этапа: • обнаружение ошибки; • поиск её местонахождения; • определение причины ошибки; • исправление ошибки. Можно выделить следующие типы ошибок: • синтаксические ошибки — обнаруживаются во время трансляции исходного кода и вызваны нарушением ожидаемой формы или структуры языка; • семантические ошибки — являются логическими и приводят к тому, что программа запускается, отработывает, но не даёт желаемого результата; • ошибки в процессе выполнения — не обнаруживаются при трансляции и вызывают прерывание выполнения программы (например, это ошибки, связанные с переполнением или делением на ноль). Второй этап — поиск местонахождения ошибки. Некоторые ошибки обнаружить довольно трудно. Лучший способ найти место в программе, где находится ошибка, это разбить программу на части и произвести их отладку отдельно друг от друга. Третий этап — выяснение причины ошибки. После определения местонахождения ошибки обычно проще определить причину неправильной работы программы. Последний этап — исправление ошибки. После этого при повторном запуске программы, может обнаружиться следующая ошибка, и процесс отладки начнётся заново.

Подпрограмма — это, как правило, функционально законченный участок кода, который можно многократно вызывать из разных мест программы. В отличие от простых переходов из подпрограмм существует возврат на команду, следующую за вызовом. Если в программе встречается одинаковый участок кода, его можно

оформить в виде подпрограммы, а во всех нужных местах поставить её вызов. При этом подпрограмма будет содержаться в коде в одном экземпляре, что позволит уменьшить размер кода всей программы

3 Выполнение лабораторной работы

Записываю в lab09-1.asm программу из листинга,при этом добавив вторую подпрограмму для подсчета сложной функции. (рис. 3.1).

```
mov eax, result
call sprint
mov eax,[res]
call iprintLF

call quit

_calcul:
push ebx
mov ebx,2
mul ebx

add eax, 7

pop ebx
ret

_subcalcul:
push ebx
mov ebx, 3
mul ebx
dec ebx
mov [res],eax

pop ebx
ret
```

Рис. 3.1: Окно MidnightCommander

Проверка вывода программы (рис. 3.2).

```
[z@fedora lab09]$ nasm -f elf lab09-1.asm
[z@fedora lab09]$ ld -m elf_i386 -o lab09-1 lab09-1.o
[z@fedora lab09]$ ./lab09-1
Введите x: 5
2x+7=17
[z@fedora lab09]$
```

Рис. 3.2: запуск lab09-1

Создаю файл lab9-2.asm и проассемблируем его с другими ключами, чтобы была возможность открыть этот файл через gdb (рис. 3.3).

```
[z@fedora lab09]$ nasm -f elf -g -l lab09-2.lst lab09-2.asm
[z@fedora lab09]$ ld -m elf_i386 -o lab09-2 lab09.o
[z@fedora lab09]$ ./lab09-2
Hello, world!
[z@fedora lab09]$
```

Рис. 3.3: запуск lab09-2

Открываю файл lab9-2 с помощью gdb (рис. 3.4).

```
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.
This GDB was configured as "x86_64-pc-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://bugs.gentoo.org/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from lab9-2...
(gdb)
```

Рис. 3.4: запуск lab09-2

Поставим точку остановки на метке _start (рис. 3.5).

```
Breakpoint 1, _start () at lab9-2.asm:9
9      mov eax, 4
(gdb)
```

Рис. 3.5: окно gdb

В представлении АТТ в виде 16-ричного числа записаны первые аргументы всех команд, а в представлении intel так записываются адреса вторых аргументов. Включим режим псевдографики, с помощью которого отображается код программы и содержимое регистров(рис. 3.6).

The screenshot shows the GDB interface with the following content:

```

[ Register Values Unavailable ]

B> 0x8049000 <_start> mov eax,0x4
0x8049005 <_start+5> mov ebx,0x1
0x804900a <_start+10> mov ecx,0x804a000
0x804900f <_start+15> mov edx,0x8
0x8049014 <_start+20> int 0x80
0x8049016 <_start+22> mov eax,0x4
0x804901b <_start+27> mov ebx,0x1
0x8049020 <_start+32> mov ecx,0x804a008

native process 6166 In: _start L9 PC: 0x8049000
(gdb) layout regs
(gdb)

```

Рис. 3.6: окно gdb

Поставим точку остановки на метке _start (рис. 3.7).

The screenshot shows the GDB interface with the following content:

```

Breakpoint 1, _start () at lab9-2.asm:9
9 mov eax, 4
(gdb)

```

Рис. 3.7: окно gdb

Установим еще одну точку остановк (рис. 3.8).

The screenshot shows the GDB interface with the following content:

```

(gdb) layout regs
(gdb) info breakpoints
Num      Type           Disp Enb Address      What
1        breakpoint     keep y  0x08049000 lab9-2.asm:9
          breakpoint already hit 1 time
(gdb)

```

Рис. 3.8: окно gdb

Посмотрим информацию о наших точках останова, используя команду `i b` (рис. 3.9).

```
(gdb) b *0x8049014
Breakpoint 2 at 0x8049014: file lab9-2.asm, line 13.
(gdb) i b
Num      Type           Disp Enb Address      What
1        breakpoint     keep y   0x08049000 lab9-2.asm:9
          breakpoint already hit 1 time
2        breakpoint     keep y   0x08049014 lab9-2.asm:13
(gdb) █
```

Рис. 3.9: окно gdb

В отладчике можно вывести текущее значение переменных (рис. 3.10).

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

Рис. 3.10: окно gdb

Также можно обращаться по адресам переменных. Здесь был заменен первый символ переменной `msg2` на символ отступа (рис. 3.11).

```
(gdb) set {char}&msg2=9
(gdb) x/1sb &msg2
0x804a008 <msg2>: "\torld!\n\034"
(gdb) █
```

Рис. 3.11: окно gdb

Задаю регистру значения (рис. 3.12).

```
(gdb) set $ebx=2
(gdb) p/s $ebx
$1 = 2
(gdb) █
```

Рис. 3.12: окно gdb

Копирую файл из предыдущей лабораторной работы, переименовываю и создаю исполняемый файл. Создаю точку останова на метке `_start` и запускаю программу. (рис. 3.13).

```
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.
This GDB was configured as "x86_64-pc-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://bugs.gentoo.org/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from lab09-3...
(gdb) █
```

Рис. 3.13: окно gdb

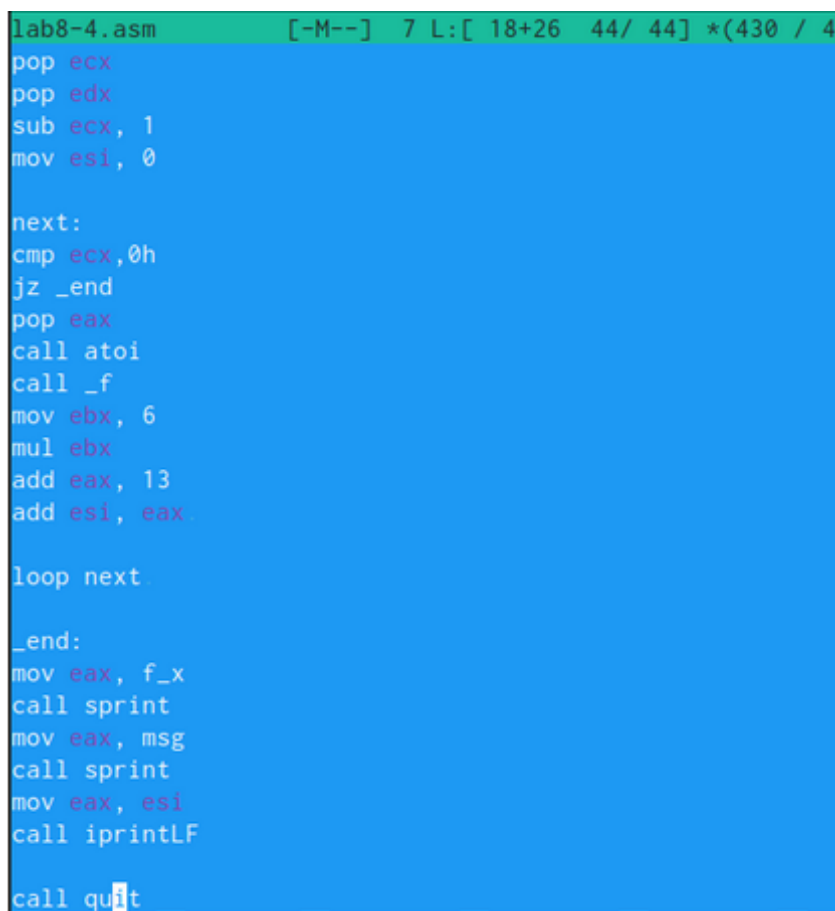
Заметим что, все остальные адреса аргументов располагаются в 4 байтах друг от друга. (рис. 3.14).

```
Breakpoint 1, _start () at lab09-3.asm:5
5      pop ecx ; Извлекаем из стека в 'ecx' количество
(gdb) x/x $esp
0xffffc2a0: 0x00000005
(gdb) x/s *(void**)(esp + 4)
```

Рис. 3.14: окно gdb

4 Самостоятельная работа

Программа из лабораторной 8, но с использованием подпрограмм. (рис. 4.1).

The image shows a screenshot of a Midnight Commander window. The title bar at the top is green and contains the text 'lab8-4.asm' followed by some status information in brackets. The main area has a blue background and displays assembly code. The code starts with 'pop ecx', 'pop edx', 'sub ecx, 1', and 'mov esi, 0'. It then enters a loop labeled 'next:' which contains 'cmp ecx, 0h', 'jz _end', 'pop eax', 'call atoi', 'call _f', 'mov ebx, 6', 'mul ebx', 'add eax, 13', and 'add esi, eax'. After the loop, it goes to '_end:' and contains 'mov eax, f_x', 'call sprint', 'mov eax, msg', 'call sprint', 'mov eax, esi', 'call iprintLF', and 'call quit'.

```
lab8-4.asm [-M--] 7 L:[ 18+26 44/ 44] *(430 / 4
pop ecx
pop edx
sub ecx, 1
mov esi, 0

next:
cmp ecx, 0h
jz _end
pop eax
call atoi
call _f
mov ebx, 6
mul ebx
add eax, 13
add esi, eax

loop next

_end:
mov eax, f_x
call sprint
mov eax, msg
call sprint
mov eax, esi
call iprintLF
call quit
```

Рис. 4.1: окно Midnightcommander

Проверка вывода программы. (рис. 4.2).

```
[z@fedora lab09]$ ./lab08-4
f(x)=6x+13
Результат: 0
[z@fedora lab09]$ ./lab08-4 2 5 6
f(x)=6x+13
Результат: 1745
[z@fedora lab09]$
```

Рис. 4.2: запуск lab08-4

5 Выводы

В ходе выполнения лабораторной работы я приобрел навыки написания программ с использованием подпрограмм. Я ознакомился с методами отладки при помощи GDB и его основными возможностями. # Список литературы{unnumbered}