Отчёт по лабараторной работе №9

Понятие подпрограммы. Отладчик GDB.

Корчаги Алексей Павлович

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Приобретение навыков 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 три этапа: • посик ошибки; • определение причины ошибки; • исправление ошибки.

GDB — отладчик проекта GNU работает на большенстве UNIX-подобных системах и может проводить процесс отладки для кода написанного на многих языках программирования. GDB предлагает обширные средства для слежения и контроля за выполнением компьютерных программ. От- ладчик не содержит собственного графического пользовательского интерфейса и использует стандартный текстовый интерфейс консоли. Однако для GDB существует несколько сторон- них графических надстроек, а кроме того, некоторые интегрированные среды разработки используют его в качестве базовой подсистемы отладки. Отладчик GDB (как и любой другой отладчик) позволяет увидеть, что происходит «внутри» программы в момент её выполнения или что делает программа в момент сбоя. GDB может выполнять следующие действия: • начать выполнение программы, задав всё, что может повлиять на её поведение; • остановить программу при указанных условиях; • исследовать, что случилось, когда программа остановилась

3 Выполнение лабораторной работы

Создал директорию и файл для выполненние лаборотной работы(рис. 3.1).

```
apkorchagin@dk5n59 ~ $ mkdir ~/work/arch-pc/lab09
apkorchagin@dk5n59 ~ $ cd ~/work/arch-pc/lab09
apkorchagin@dk5n59 ~/work/arch-pc/lab09 $ touch lab9-1.asm
apkorchagin@dk5n59 ~/work/arch-pc/lab09 $
```

Рис. 3.1: Созданние файла

Записал код в файл lab9-1.asm(рис. 3.2).

```
%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
```

Рис. 3.2: Редактированние файла

Проверил работу программы(рис. 3.3).

```
apkorchagin@dk5n59 ~/work/arch-pc/lab09 $ nasm -f elf lab9-1.asm
apkorchagin@dk5n59 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-1 lab9-1.o
apkorchagin@dk5n59 ~/work/arch-pc/lab09 $ ./lab9-1
Введите x: 5
2x+7=17
apkorchagin@dk5n59 ~/work/arch-pc/lab09 $
```

Рис. 3.3: Работа программы

(рис. **??**).

Создал файл lab9-2.asm(рис. 3.4).

```
apkorchagin@dk5n59 ~/work/arch-pc/lab09 $ touch lab9-2.asm
apkorchagin@dk5n59 ~/work/arch-pc/lab09 $
```

Рис. 3.4: Созданние файла

Записал код в файл(рис. 3.5).

```
 1 SECTION .data
 2 msg1: db "Hello, ",0x0
 3 msg1Len: equ $ - msg1
 4 msg2: db "world!",0xa
 5 msg2Len: equ $ - msg2
 6 SECTION .text
 7 global _start
 8 _start:
 9 mov eax, 4
10 mov ebx, 1
11 mov ecx, msg1
12 mov edx, msg1Len
13 int 0x80
14 mov eax, 4
15 mov ebx, 1
16 mov ecx, msg2
17 mov edx, msg2Len
18 int 0x80
19 mov eax, 1
20 mov ebx, 0
21 int 0x80
```

Рис. 3.5: Редактированние файла

Ассемблирол файл lab9-2.asm, чтобы открыть файл через gdb(рис. 3.6).

```
apkorchagin@dk5n59 ~/work/arch-pc/lab09 $ nasm -f elf -g -l lab9-2.lst lab9-2.asm
apkorchagin@dk5n59 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-2 lab09-2.o
ld: невозможно найти lab09-2.o: Нет такого файла или каталога
apkorchagin@dk5n59 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-2 lab9-2.o
apkorchagin@dk5n59 ~/work/arch-pc/lab09 $ gdb lab9-2
```

Рис. 3.6: Работа файла

Открыл файл lab9-2.asm через gdb(рис. 3.7).

```
apkorchagin@dk5n59 ~/work/arch-pc/lab09 $ gdb lab9-2
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.
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) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/a/p/apkorchagin/work/arch-pc/lab09/

Hello, world!
[Inferior 1 (process 8537) exited normally]
```

Рис. 3.7: Работа с gdb

Поставил breakpoint на _start(рис. 3.8).

```
(gdb) break _start
Breakpoint 1 at 0x8049000: file lab9-2.asm, line 9.
(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/a/p/apkorchagin/work/arch-pc/lab09/lab9-2


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

Рис. 3.8: Установка breakpoint

Начал просмотр дизассемблированного кода с синтаксисом intel(рис. 3.9).

```
(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. 3.9: Дизассемблированный код

В синтаксисе ATT в виде 16-ти ричного числа записаны первые аргументы всех комманд, в синтаксисе Intel так записываются адреса вторых аргументов. запустим режим псевдографики, с помощью которого отображается код программы и содержимое регистров(рис. 3.10).

```
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 8540 In: _start                          L9    PC: 0x8049000
(gdb) layout regs
(gdb)
```

Рис. 3.10: Содержанние регистров

С помощью i b посмотрим информацию о точках остановки(рис. 3.11).

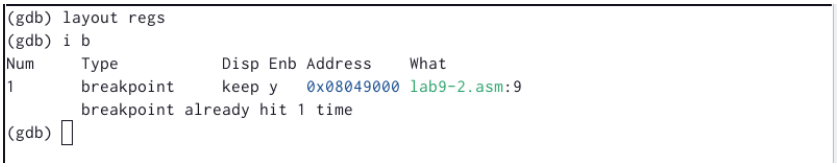

Рис. 3.11: Команда i b

Добавил ещё одну точку остановки по адресу(рис. 3.12).

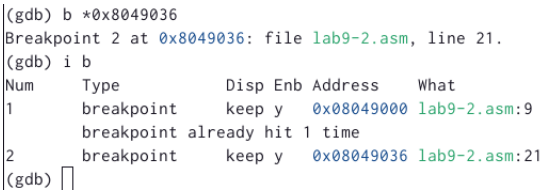

Рис. 3.12: Взаимодействие с адресом

С помощью команды i r вывел значенние регистров(рис. 3.13).

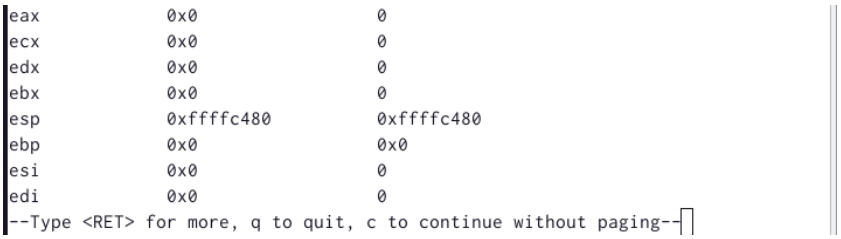

Рис. 3.13: Значенние регистров

Вывел значение переменной по именни(рис. 3.14).

Рис. 3.14: Значенние переменной msg1

Изменил значенние переменной msg1(рис. 3.15).

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

Рис. 3.15: Измененние значения msg1

Изменил значенние переменной msg2(рис. 3.16).

```
(gdb) set {char}&msg2=8
(gdb) x/lsb &msg2
0x804a008 <msg2>:       "\borld!\n\034"
(gdb)
```

Рис. 3.16: Измененние значения msg2

Вывел значение регистра ebx, в трёх разных форматах: в строчном(p/s); в 16-речном(p/x);в двоичном(p/t)(рис. 3.17).

```
(gdb) p/s $edx
$1 = 0
(gdb) p/x
$2 = 0x0
(gdb) p/t
$3 = 0
(gdb)
```

Рис. 3.17: Значенние регистра

Задал значенние регистру, при попытке задания регустру строкового занчения возникает ошибка(рис. 3.18).

```
(gdb) set $ebx="2"
evaluation of this expression requires the program to have a function "malloc".
(gdb) set $ebx=2
(gdb) p/s $ebx
$5 = 2
(gdb)
```

Рис. 3.18: Присваивание занченния регистру

Скопировал из 8 лабороторной работы файл переименовал и создал исполняймый файл. Открыл файл создал точку остановки(рис. 3.19).

```
apkorchagin@dk5n59 ~ $ cp ~/work/arch-pc/lab08/lab8-2.asm ~/work/arch-pc/lab09/lab9-3.asm
apkorchagin@dk5n59 ~ $ cd ~/work/arch-pc/lab09
apkorchagin@dk5n59 ~/work/arch-pc/lab09 $ nasm -f elf -g -l lab9-3.lst lab9-3.asm
apkorchagin@dk5n59 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-3 lab9-3.o
apkorchagin@dk5n59 ~/work/arch-pc/lab09 $ gdb --args lab9-3 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.
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 lab9-3...
(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab9-3.asm, line 5.
(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/a/p/apkorchagin/work/arch-pc/lab09/lab9-3 аргумент1 арг
умент 2 аргумент\ 3

Breakpoint 1, _start () at lab9-3.asm:5
5       pop ecx ; Извлекаем из стека в `ecx` количество
```

Рис. 3.19: Копирование файла

Посмотерл на содержимое того, что находится по адресу регистра esp(рис. 3.20).

```
(gdb) x/x $esp
0xffffc440:     0x00000005
(gdb)
```

Рис. 3.20: Значенние esp

Посмотрел на все рстальные элементы стека. Их адреса находятся на растоянии 4 байта, именно столько занимает один элемент стека(рис. **??**).

```
(gdb) x/s *(void**)($esp + 8)
0xffffc6ec:     "аргумент1"
(gdb) x/s *(void**)($esp + 4)
0xffffc6a5:     "/afs/.dk.sci.pfu.edu.ru/home/a/p/apkorchagin/work/arch-pc/lab09/lab9-3"
(gdb) x/s *(void**)($esp + 12)
0xffffc6fe:     "аргумент"
(gdb) x/s *(void**)($esp + 16)
0xffffc70f:     "2"
(gdb) x/s *(void**)($esp + 20)
0xffffc711:     "аргумент 3"
(gdb) x/s *(void**)($esp + 24)
0x0:    <error: Cannot access memory at address 0x0>
(gdb)
```

4 Выполенние самостоятельной работы

(рис. **??**).

(рис. **??**).

(рис. **??**).

(рис. **??**).

(рис. **??**).

(рис. **??**).

(рис. **??**).

(рис. **??**).

5 Выводы

В ходе выполненния работа я приобрёл навыки написания программ с использованием подпрограмм и познакомился с методами отладки при помощи GDB.