

Лабораторная работа No10 .

Понятие подпрограммы. Отладчик GDB.

Георгий Никифоров Сергеевич

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выводы	14
	Список литературы	15

Список иллюстраций

2.1	Листинг 10.1	6
2.2	Программа из листинга 9.1	7
2.3	Запуск программы	7
2.4	Запуск программы	7
2.5	Работа программы	8
2.6	layout regs	8
2.7	layout regs	9
2.8	Просмотр точек останова	9
2.9	Значения регистров	9
2.10	Строка msg1	10
2.11	Строка msg2	10
2.12	Замена букв в msg1	10
2.13	Замена букв в msg2	10
2.14	Значение регистра edx	11
2.15	Изменение регистра edx	11
2.16	Количество аргументов в программе	11
2.17	Смещение на кол-во байт	12
2.18	Функция как подпрограмма	12
2.19	Запуск программы	13

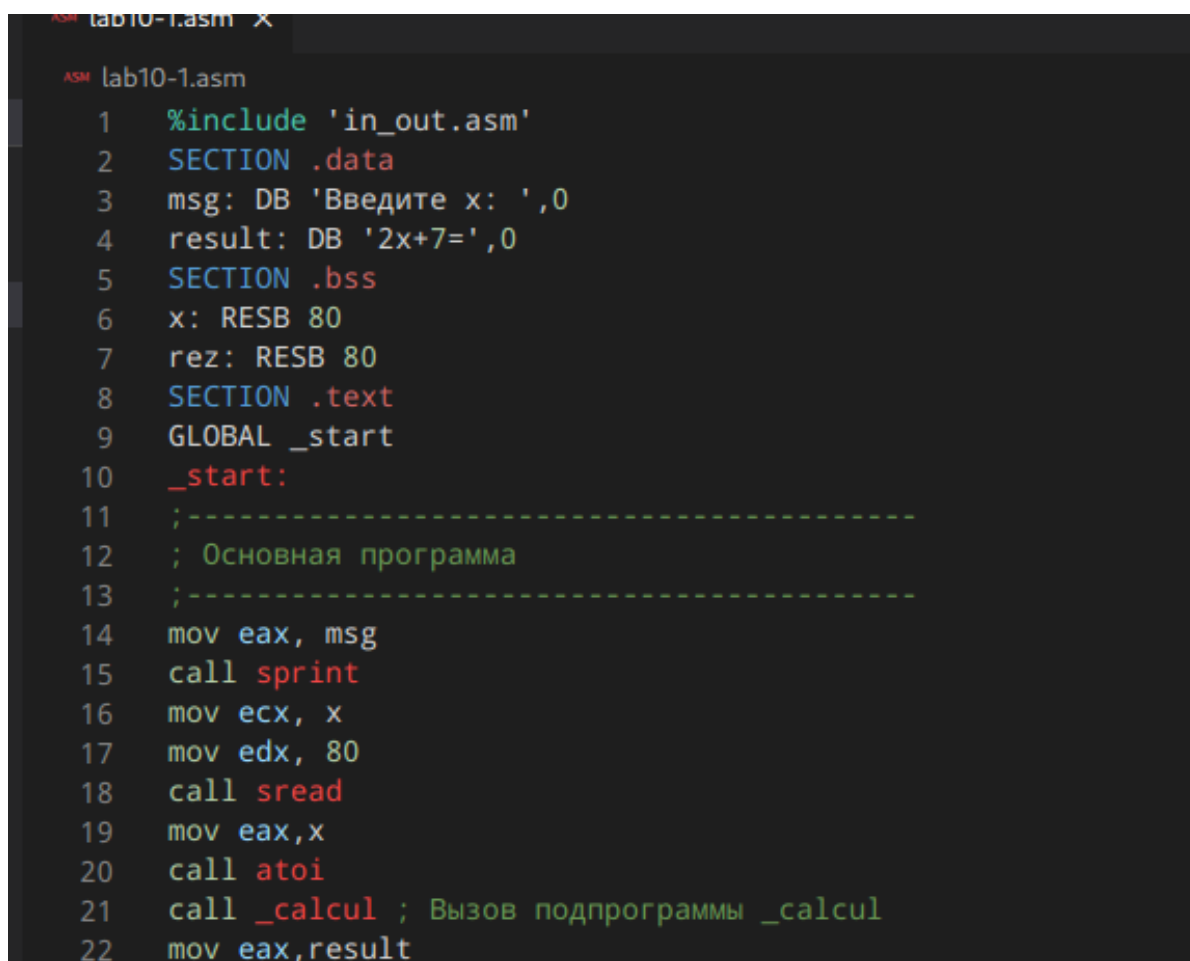
Список таблиц

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

Листинг первой программы



```
lab10-1.asm
ASM lab10-1.asm
1  %include 'in_out.asm'
2  SECTION .data
3  msg: DB 'Введите x: ',0
4  result: DB '2x+7=',0
5  SECTION .bss
6  x: RESB 80
7  rez: RESB 80
8  SECTION .text
9  GLOBAL _start
10 _start:
11 ;-----
12 ; Основная программа
13 ;-----
14 mov eax, msg
15 call sprint
16 mov ecx, x
17 mov edx, 80
18 call sread
19 mov eax, x
20 call atoi
21 call _calcul ; Вызов подпрограммы _calcul
22 mov eax, result
```

Рис. 2.1: Листинг 10.1

Работа программы из листинг 10.1

```

gsnikiforov@dk3n31 ~/work/arch-pc/lab10 $ nasm -f elf lab10-1.
lab10-1.asm:24: error: symbol `res' not defined
lab10-1.asm:34: error: symbol `rez' not defined
gsnikiforov@dk3n31 ~/work/arch-pc/lab10 $ nasm -f elf lab10-1.
gsnikiforov@dk3n31 ~/work/arch-pc/lab10 $ ld -m elf_i386 -o la
gsnikiforov@dk3n31 ~/work/arch-pc/lab10 $ ./lab10-1
Введите x: 3
2x+7=13
gsnikiforov@dk3n31 ~/work/arch-pc/lab10 $ 

```

Рис. 2.2: Программа из листинга 9.1

Преобразование программы из листинга 10.1

```

gsnikiforov@dk2n26 ~/work/arch-pc/lab10 $ ./lab10-1
Введите x: 2
2 * (3x - 1) + 7 = 17
gsnikiforov@dk2n26 ~/work/arch-pc/lab10 $ 

```

lab10-1.asm - lab10 - Visual Studi... report.md - Visual Studio Code lab10-1.asm (/tm

Рис. 2.3: Запуск программы

Работа программы из листинга 10.2

```

(gdb) r
Starting program: /afs/.dk.sci.pfu.edu.ru/home/g/s/gsnikiforov/work/arch-pc/la
Hello, World!
[Inferior 1 (process 5368) exited normally]
(gdb) 

```

Рис. 2.4: Запуск программы

Установка брейкпоинта

```
Breakpoint 1 at 0x8049000: file lab10-2.asm, line 10.
(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/g/s/gsnikiforov/work/arch-pc/la
Breakpoint 1, _start () at lab10-2.asm:10
10      mov eax, 4
(gdb) █
```

Рис. 2.5: Работа программы

Окно регистров

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
	0x8049016	<_start+22>	mov	\$0x4,%eax
	0x804901b	<_start+27>	mov	\$0x1,%ebx
	0x8049020	<_start+32>	mov	\$0x804a008,%ecx
	0x8049025	<_start+37>	mov	\$0x7,%edx
	0x804902a	<_start+42>	int	\$0x80
	0x804902c	<_start+44>	mov	\$0x1,%eax
	0x8049031	<_start+49>	mov	\$0x0,%ebx
	0x8049036	<_start+54>	int	\$0x80
	0x8049038		add	%al,(%eax)

Рис. 2.6: layout regs


```

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
0x8049016 <_start+22>    mov     $0x4,%eax
0x804901b <_start+27>    mov     $0x1,%ebx
0x8049020 <_start+32>    mov     $0x804a008,%ecx
0x8049025 <_start+37>    mov     $0x7,%edx
0x804902a <_start+42>    int     $0x80
0x804902c <_start+44>    mov     $0x1,%eax
0x8049031 <_start+49>    mov     $0x0,%ebx
0x8049036 <_start+54>    int     $0x80

exec No process In:
warning: Source file is more recent than executable.
(gdb)

```

Рис. 2.7: layout regs

Точки остановки

```

(gdb) i b
Num      Type      Disp Enb Address      What
      breakpoint keep y  0x08049000 lab10-2.asm:10
(gdb)

```

Рис. 2.8: Просмотр точек остановки

Информация о регистрах

```

Register group: general
eax      0x0      0      ecx      0x0      0
ebx      0x0      0      esp      0xffffc380  0xffffc380
esi      0x0      0      edi      0x0      0
eflags   0x202    [ IF ]   cs       0x23      35
ds       0x2b     43      es       0x2b      43
gs       0x0      0

```

Рис. 2.9: Значения регистров

Значение памяти по нужному адресу

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

Рис. 2.10: Строка msg1

Обращение к памяти через адрес

```
Breakpoint 1, _start () at lab10-2.asm:10
(gdb) x/1sb 0x804a008
0x804a008 <msg2>:      "World!\n\034"
(gdb) █
```

Рис. 2.11: Строка msg2

Использование команды set

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

Рис. 2.12: Замена букв в msg1

Использование команды set

```
(gdb) set {char}&msg2='B'
(gdb) x/1sb &msg2
0x804a008 <msg2>:      "Bor1d!\n\034"
(gdb) █
```

Рис. 2.13: Замена букв в msg2

Значение регистра edx в разных форматах

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

Рис. 2.14: Значение регистра edx

Изменение значения регистра на '2' и на 2

```
(gdb) set $ebx='2'
(gdb) p/s $ebx
$4 = 50
(gdb) set $ebx='2
Unmatched single quote.
(gdb) set $ebx=2
(gdb) p/s $ebx
$5 = 2
```

Рис. 2.15: Изменение регистра edx

Программа 10-3.asm

```
breakpoint 1 at 0x80490e8. File lab10-3.asm,
(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/ho

Breakpoint 1, _start () at lab10-3.asm:5
5      pop ecx ; Извлекаем из стека в `ecx`
(gdb) x/x $esp
0xffffc330: 0x00000005
(gdb)
```

Рис. 2.16: Количество аргументов в программе

Адрес и смещение аргументов программы

```

(gdb) x/s *(void**)(esp + 4)
0xffffc5cf:      "/afs/.dk.sci.pfu.edu.ru/home/g/s
(gdb) x/s *(void**)(esp + 8)
0xffffc617:      "аргумент1"
(gdb) x/s *(void**)(esp + 12)
0xffffc629:      "аргумент"
(gdb) x/s *(void**)(esp + 16)
0xffffc63a:      "2"
(gdb) x/s *(void**)(esp + 20)
0xffffc63c:      "аргумент 3"
(gdb) x/s *(void**)(esp + 24)
0x0:      <error: Cannot access memory at address 0

```

Рис. 2.17: Смещение на кол-во байт

Самостоятельная работа

Преобразование программы из лр 9

```

snikiforov@dk2n26 ~/work/arch-pc/lab10 $ ./lab10-4 1 2
(x) = 15 * x + 2
ошибка сегментирования (стек памяти сброшен на диск)
snikiforov@dk2n26 ~/work/arch-pc/lab10 $ nasm -f elf -g -l lab10-4.lst lab10-4
snikiforov@dk2n26 ~/work/arch-pc/lab10 $ ld -m elf_i386 -o lab10-4 lab10-4.o
snikiforov@dk2n26 ~/work/arch-pc/lab10 $ ./lab10-4 1 2
(x) = 15 * x + 2
результат: 49
snikiforov@dk2n26 ~/work/arch-pc/lab10 $

```

Рис. 2.18: Функция как подпрограмма

Исправил программу 10.3

```
f(x) = 15 * x + 2
Результат: 49
gsnikiforov@dk2n26 ~/work/arch-pc/lab10 $ touch lab10-5.asm
gsnikiforov@dk2n26 ~/work/arch-pc/lab10 $ nasm -f elf -g -l lab10-5.lst lab10-
gsnikiforov@dk2n26 ~/work/arch-pc/lab10 $ ld -m elf_i386 -o lab10-5 lab10-5.o
gsnikiforov@dk2n26 ~/work/arch-pc/lab10 $ ./lab10-5
Результат: 25
gsnikiforov@dk2n26 ~/work/arch-pc/lab10 $ █
```

Рис. 2.19: Запуск программы

3 Выводы

Приобрел навыки написания программ с использованием подпрограмм. Ознакомился с методами отладки при помощи GDB и его основными возможностями.

Список литературы