

Отчет по лабораторной работе №9

Архитектура компьютера

Никифоров Захар Сергеевич

Содержание

1 Цель работы	5
2 Порядок выполнения работы	6
2.1 Реализация подпрограмм в NASM	6
2.2 Отладка программ с помощью GDB	7
2.3 Добавление точек останова	9
2.4 Работа с данными программы в GDB	11
2.5 Обработка аргументов командной строки в GDB	14
3 Задание для самостоятельной работы	16
4 Выводы	20

Список иллюстраций

2.1	Скриншот терминала тс 1	6
2.2	Скриншот терминала тс 2	6
2.3	Скриншот терминала тс 3	7
2.4	Скриншот терминала тс 4	7
2.5	Скриншот терминала тс 5	8
2.6	Скриншот терминала тс 6	8
2.7	Скриншот терминала тс 7	9
2.8	Скриншот терминала тс 8	10
2.9	Скриншот терминала тс 9	11
2.10	Скриншот терминала тс 10	12
2.11	Скриншот терминала тс 11	13
2.12	Скриншот терминала тс 12	14
2.13	Скриншот терминала тс 13	14
2.14	Скриншот терминала тс 14	15
3.1	Скриншот терминала тс 15	17
3.2	Скриншот терминала тс 16	18
3.3	Скриншот терминала тс 17	18
3.4	Скриншот терминала тс 18	19

Список таблиц

1 Цель работы

Приобретение навыков написания программ с использованием подпрограмм.
Знакомство с методами отладки при помощи GDB и его основными возможностями.

2 Порядок выполнения работы

2.1 Реализация подпрограмм в NASM

Создаем каталог *lab09*, а в нем создаем файл *lab9-1.asm*, записываем код из *Листинг 8.1* и компилируем его.

The screenshot shows the mc file manager interface. The title bar says "mc [zachnickif@fedora]:~/work/arch-pc/lab09". The left pane shows the directory structure: . и .. in the current directory (~/work/arch-pc/lab09). The right pane lists files with columns: Имя (Name), Размер (Size), Дата (Date), and Правки (Permissions). The files listed are: in_out.asm (3942, ноя 8 13:51), *lab9-1 (5056, дек 7 17:32), lab9-1.asm (493, дек 7 17:31), and lab9-1.o (1456, дек 7 17:31). The "lab9-1.asm" file is selected.

Левая панель	Файл	Команда	Настройки
<-- ~/work/arch-pc/lab09		-----.[^]>-	
.и	Имя	Размер	Дата правки
/..		-ВВЕРХ-	дек 7 17:20
	in_out.asm	3942	ноя 8 13:51
	*lab9-1	5056	дек 7 17:32
	lab9-1.asm	493	дек 7 17:31
	lab9-1.o	1456	дек 7 17:31

Рисунок 2.1: Скриншот терминала mc 1

The screenshot shows a terminal window with the command "zachnickif@fedora:~/work/arch-pc/lab09\$./lab9-1" entered. The output is: "Введите x: 4" followed by "2x+7=15".

```
zachnickif@fedora:~/work/arch-pc/lab09$ ./lab9-1
Введите x: 4
2x+7=15
```

Рисунок 2.2: Скриншот терминала mc 2

Получаем вывод результата функции. Запишем в код подпрограмму *_subcalcul*, вычисляющую $f(g(x))$, снова соберем и проверим вывод.

```
_calcul:  
<----->call _subcalcul  
<----->mov ebx, 2  
<----->mul ebx  
<----->add eax, 7  
<----->mov [res], eax  
<----->ret  
  
_subcalcul:  
<----->mov ebx, 3  
<----->mul ebx  
<----->sub eax, 1  
<----->ret
```

Рисунок 2.3: Скриншот терминала mc 3

```
zachnickif@fedora:~/work/arch-pc/lab09$ ./lab9-1  
Введите x: 5  
f(g(x))=35
```

Рисунок 2.4: Скриншот терминала mc 4

Мы видим, что программа работает корректно.

2.2 Отладка программам с помощью GDB

Создадим файл *lab9-2.asm*, запишем в него текст из *Листинг 9.2*, скомпилируем с получением файла трансляции и запустим в *gdb*.

```
Debuginfod has been enabled.  
To make this setting permanent, add '  
Downloading separate debug info for $  
Hello, world!  
[Inferior 1 (process 5240) exited normally]
```

Рисунок 2.5: Скриншот терминала mc 5

Программа работает, как надо, перейдем к её анализу. Поставим брейкпоинт на старте программы.

```
Breakpoint command: m -t i386 -nsp  
(gdb) break _start  
Breakpoint 1 at 0x8048080: file lab9-2.asm, line 9.  
(gdb) run  
Starting program: /home/vernel/work/arch-pc/lab09/lab9-2  
  
Breakpoint 1, _start () at lab9-2.asm:9  
9          mov eax, 4  
(gdb)
```

Рисунок 2.6: Скриншот терминала mc 6

Посмотрим на диассимилированный код в двух синтаксисах.

```
(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x08048080 <+0>:    mov    $0x4,%eax
    0x08048085 <+5>:    mov    $0x1,%ebx
    0x0804808a <+10>:   mov    $0x8049000,%ecx
    0x0804808f <+15>:   mov    $0x8,%edx
    0x08048094 <+20>:   int    $0x80
    0x08048096 <+22>:   mov    $0x4,%eax
    0x0804809b <+27>:   mov    $0x1,%ebx
    0x080480a0 <+32>:   mov    $0x8049008,%ecx
    0x080480a5 <+37>:   mov    $0x7,%edx
    0x080480aa <+42>:   int    $0x80
    0x080480ac <+44>:   mov    $0x1,%eax
    0x080480b1 <+49>:   mov    $0x0,%ebx
    0x080480b6 <+54>:   int    $0x80
End of assembler dump.
(gdb) set disassembly-flavor intel
(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x08048080 <+0>:    mov    eax,0x4
    0x08048085 <+5>:    mov    ebx,0x1
    0x0804808a <+10>:   mov    ecx,0x8049000
    0x0804808f <+15>:   mov    edx,0x8
    0x08048094 <+20>:   int    0x80
    0x08048096 <+22>:   mov    eax,0x4
    0x0804809b <+27>:   mov    ebx,0x1
    0x080480a0 <+32>:   mov    ecx,0x8049008
    0x080480a5 <+37>:   mov    edx,0x7
    0x080480aa <+42>:   int    0x80
    0x080480ac <+44>:   mov    eax,0x1
    0x080480b1 <+49>:   mov    ebx,0x0
    0x080480b6 <+54>:   int    0x80
End of assembler dump.
```

Рисунок 2.7: Скриншот терминала mc 7

Мы можем заметить разницу в отображении операндов инструкций в разных синтаксисах.

2.3 Добавление точек останова

Посмотрим на вид в псевдографике и список брейкпоинтов.

```
Register group: general
ebp      0x0          0x0
esi      0x0          0
edi      0x0          0
eip      0x8048080    0x8048080 <_start>
eflags   0x202        [ IF ]
cs       0x23         35
ss       0x2b         43
ds       0x2b         43
es       0x2b         43
fs       0x0          0
gs       0x0          0

0x8048f44  add   BYTE PTR [eax],al
0x8048f46  add   BYTE PTR [eax],al
0x8048f48  add   BYTE PTR [eax],al
0x8048f4a  add   BYTE PTR [eax],al
0x8048f4c  add   BYTE PTR [eax],al
0x8048f4e  add   BYTE PTR [eax],al
0x8048f50  add   BYTE PTR [eax],al
0x8048f52  add   BYTE PTR [eax],al
0x8048f54  add   BYTE PTR [eax],al
0x8048f56  add   BYTE PTR [eax],al
0x8048f58  add   BYTE PTR [eax],al

native process 5376 (asm) In: _start
(gdb) layout regs
(gdb) i b
Num  Type      Disp Enb Address  What
1    breakpoint keep y  0x08048080 lab9-2.asm:9
      breakpoint already hit 1 time
(gdb)
```

Рисунок 2.8: Скриншот терминала mc 8

Установим новый брейпойнт по адресу.

```
b+ 0x8048080 <_start>    mov    $0x4,%eax
0x8048085 <_start+5>    mov    $0x1,%ebx
0x804808a <_start+10>   mov    $0x8049000,%ecx
0x804808f <_start+15>   mov    $0x8,%edx
0x8048094 <_start+20>   int    $0x80
0x8048096 <_start+22>   mov    $0x4,%eax
0x804809b <_start+27>   mov    $0x1,%ebx
0x80480a0 <_start+32>   mov    $0x8049008,%ecx
0x80480a5 <_start+37>   mov    $0x7,%edx
0x80480aa <_start+42>   int    $0x80
0x80480ac <_start+44>   mov    $0x1,%eax
b+ 0x80480b1 <_start+49>   mov    $0x0,%ebx
0x80480b6 <_start+54>   int    $0x80

exec No process (asm) In:                                L??  PC: ???
(gdb) b _start
Breakpoint 1 at 0x8048080: file lab9-2.asm, line 9.
(gdb) b 0x80480b1
Function "0x80480b1" not defined.
Make breakpoint pending on future shared library load? (y or [n]) n
(gdb) b *0x80480b1
Breakpoint 2 at 0x80480b1: file lab9-2.asm, line 22.
(gdb) 
```

Рисунок 2.9: Скриншот терминала mc 9

2.4 Работа с данными программы в GDB

Посмотрим содержимое регистров

```

B+ 0x8048080 <_start>    mov    $0x4,%eax
0x8048085 <_start+5>    mov    $0x1,%ebx
0x804808a <_start+10>   mov    $0x8049000,%ecx
0x804808f <_start+15>   mov    $0x8,%edx
0x8048094 <_start+20>   int    $0x80
>0x8048096 <_start+22>  mov    $0x4,%eax
0x804809b <_start+27>  mov    $0x1,%ebx
0x80480a0 <_start+32>  mov    $0x8049008,%ecx
0x80480a5 <_start+37>  mov    $0x7,%edx
0x80480aa <_start+42>  int    $0x80
0x80480ac <_start+44>  mov    $0x1,%eax
b+ 0x80480b1 <_start+49>  mov    $0x0,%ebx
0x80480b6 <_start+54>  int    $0x80
0x80480b8          add    %al,(%eax)
0x80480ba          add    %al,(%eax)
0x80480bc          add    %al,(%eax)
0x80480be          add    %al,(%eax)
0x80480c0          add    %al,(%eax)
0x80480c2          add    %al,(%eax)
0x80480c4          add    %al,(%eax)
0x80480c6          add    %al,(%eax)
0x80480c8          add    %al,(%eax)
0x80480ca          add    %al,(%eax)

native process 5663 (asm) In: _start                                L15  PC: 0x804
eax          0x8          8
ecx          0x8049000     134516736
edx          0x8          8
ebx          0x1          1
esp          0xfffffcf20  0xfffffcf20
ebp          0x0          0x0
esi          0x0          0
edi          0x0          0
eip          0x8048096    0x8048096 <_start+22>
eflags        0x202       [ IF ]
cs           0x23         35
ss           0x2b         43
--Type <RET> for more, q to quit, c to continue without paging--■

```

Рисунок 2.10: Скриншот терминала mc 10

Теперь посмотрим содержание переменных *msg1* и *msg2* по их адрессам.

```

B+ 0x8048080 <_start>      mov    $0x4,%eax
0x8048085 <_start+5>      mov    $0x1,%ebx
0x804808a <_start+10>     mov    $0x8049000,%ecx
0x804808f <_start+15>     mov    $0x8,%edx
0x8048094 <_start+20>     int    $0x80
>0x8048096 <_start+22>     mov    $0x4,%eax
0x804809b <_start+27>     mov    $0x1,%ebx
0x80480a0 <_start+32>     mov    $0x8049008,%ecx
0x80480a5 <_start+37>     mov    $0x7,%edx
0x80480aa <_start+42>     int    $0x80
0x80480ac <_start+44>     mov    $0x1,%eax
b+ 0x80480b1 <_start+49>     mov    $0x0,%ebx
0x80480b6 <_start+54>     int    $0x80
0x80480b8          add    %al,(%eax)
0x80480ba          add    %al,(%eax)
0x80480bc          add    %al,(%eax)
0x80480be          add    %al,(%eax)
0x80480c0          add    %al,(%eax)
0x80480c2          add    %al,(%eax)
0x80480c4          add    %al,(%eax)
0x80480c6          add    %al,(%eax)
0x80480c8          add    %al,(%eax)
0x80480ca          add    %al,(%eax)

native process 5689 (asm) In: _start
eflags      0x202          [ IF ]
cs          0x23           35
ss          0x2b           43
--Type <RET> for more, q to quit, c to continue without paging
ds          0x2b           43
es          0x2b           43
fs          0x0            0
gs          0x0            0
(gdb) x/1sb &msg1
0x8049000 <msg1>:      "Hello, "
(gdb) x/1sb 0x8049008
0x8049008 <msg2>:      "world!\n\034"
(gdb) █

```

Рисунок 2.11: Скриншот терминала mc 11

П опробуем поменять символ в *msg2*. „w“ на „W“.

```
(gdb) p/s $msg1  
$0x8049000 <msg1>:      "Hello, "  
(gdb) set {char}&msg1='h'  
(gdb) x/1sb &msg1  
$0x8049000 <msg1>:      "hello, "  
(gdb) set {char}&msg2='W'  
(gdb) x/1sb &msg2  
$0x8049008 <msg2>:      "World!\n\034"
```

Рисунок 2.12: Скриншот терминала mc 12

Разница вывода команд *p/s \$ebx* связана с тем, что сначала мы присвоили „2“, который имеет значение 50 в ASCII коде, а потом число 2, которому соответствует 2.

2.5 Обработка аргументов командной строки в GDB

Копируем файл *lab8-2.asm* с новым именем *lab9-3.asm*, компилируем и запускаем его с gdb.

*lab9-3	5592	дек 7 18:27
lab9-3.asm	501	ноя 30 18:13
lab9-3.lst	12548	дек 7 18:26
lab9-3.o	2496	дек 7 18:26

Рисунок 2.13: Скриншот терминала mc 13

```
Breakpoint 1, _start () at lab9-3.asm:10
10          mov eax, msg1
(gdb) x/x $esp
0xfffffec0:    0x00000005
(gdb) x/s *(void**)( $esp + 4)
0xfffffd0a7:    "/home/vernel/work/arch-pc/lab09/lab9-3"
(gdb) x/s *(void**)( $esp + 8)
0xfffffd0ce:    "аргумент1"
(gdb) x/s *(void**)( $esp + 12)
0xfffffd0e0:    "аргумент"
(gdb) x/s *(void**)( $esp + 16)
0xfffffd0f1:    "2"
(gdb) x/s *(void**)( $esp + 20)
0xfffffd0f3:    "аргумент 3"
(gdb) x/s *(void**)( $esp + 24)
0x0:    <error: Cannot access memory at address 0x0>
(gdb)
```

Рисунок 2.14: Скриншот терминала mc 14

Адрес аргументов увеличивается с шагом 4, так как каждый указатель занимает 4 байта.

3 Задание для самостоятельной работы

1. Копируем файл *self.asm* из прошлой лабораторной, записав вычисление функции в виде подпрограммы *_calc*, компилируем и запускаем.

```
/home/vernel/work/arch-pc/lab09/self.asm
%include 'in_out.asm'
SECTION .data
    msg1 db 'Функция: f(x) = 2x + 15', 10, 0
    msg2 db 'Результат: ', 0
SECTION .bss
    buf: RESB 80
SECTION .text
global _start
_start:
    pop ecx
    pop edx
    sub ecx, 1
    mov esi, 0
next:
    cmp ecx, 0
    jz _end

    pop ebx
    mov eax, ebx
    call atoi
    call _calc

    add esi, eax

    loop next
_end:
    mov eax, msg1
    call sprint

    mov eax, msg2
    call sprint

    mov eax, esi
    call iprintLF

    mov ecx, buf
    mov edx, 80
    call sread
    call quit
_calc:
    shl eax, 1
    add eax, 15
    ret
```

Рисунок 3.1: Скриншот терминала mc 15

```
zachnickif@fedora:~/work/arch-pc/lab09$ ./self 12
Функция: f(x) = 2x + 15
Результат: 39
```

Рисунок 3.2: Скриншот терминала mc 16

Программа работает корректно

2. В программе допущена ошибка, когда передаем в `mul ebx`, мы умножаем не `ebx` на 4, а `esx` на 4, получаем 8, которая после не задействуется, поэтому выходит, что `ebx+ebx=5+5=10`, а не 25. Исправим программу.

```
self2.asm [-M--] 17 L:[ 1+14 15/
%include 'in_out.asm'
SECTION .data
div: DB 'Результат: ', 0

SECTION .text
GLOBAL _start
_start:
<----->mov eax, 3
<----->add eax, 2
<----->
<----->mov ebx, 4
<----->mul ebx
<----->
<----->add eax, 5
<----->mov edi, eax
<----->
<----->mov eax, div
<----->call sprint
<----->
<----->mov eax, edi
<----->call iprintLF
<----->
<----->call quit
```

Рисунок 3.3: Скриншот терминала mc 17

```
zachnickif@fedora:~/work/arch-pc/lab09$ ./self2  
Результат: 25
```

Рисунок 3.4: Скриншот терминала mc 18

Теперь программа работает корректно.

4 Выводы

Были приобретены навыки написания программ с использованием подпрограмм. Познакомились с методами отладки при помощи GDB и его основными возможностями.