

Отчет по лабораторной работе №9

Дисциплина: архитектура компьютера

Желобицкая П.А.

Содержание

1	Цель работы	6
2	Выполнение лабораторной работы	7
3	Выполнение заданий самостоятельной работы	20
4	Выводы	24
	Список литературы	25

Список иллюстраций

2.1	Создание каталога и файла	7
2.2	Ввод текста программы	8
2.3	Запуск и проверка файла	8
2.4	Изменения текста программы	9
2.5	Запуск и проверка файла	9
2.6	Создание файла	9
2.7	Ввод программы	10
2.8	Загрузка в отладчик gdb и запуск программы в оболочке GDB . .	10
2.9	Установка брейкпоинта	10
2.10	Запуск программы	11
2.11	Просмотр дисассимилированного кода программы	11
2.12	Переключение на отображение команд с Intel'овским синтаксисом	12
2.13	Режим псевдографики	12
2.14	Режим псевдографики	13
2.15	Проверка точки останова	13
2.16	Адрес предпоследней инструкции	13
2.17	Информация о всех точках останова	14
2.18	Содержимое регистров	14
2.19	Значение переменной msg1 по имени	14
2.20	Значение переменной msg2 по адресу	14
2.21	Изменение первого символа переменной msg1	15
2.22	Замена второго символа в переменной msg2	15
2.23	Выведение значений регистра ebx	16
2.24	Изменение значений регистра ebx	17
2.25	Копирование и создание исполняемого файла	17
2.26	Загрузка исполняемого файла в отладчик	18
2.27	Запуск программы	18
2.28	Адрес вершины стека	18
2.29	Остальные позиции стека	19
3.1	Редактирование файла	20
3.2	Создание исполняемого файла и проверка его	20
3.3	Создание файла	21
3.4	Ввод текста программы	21
3.5	Запуск программы в отладчике	21
3.6	Действия в отладчике	22
3.7	Изменения в программе	22

3.8	Запуск и проверка программы	23
3.9	Название рисунка	23

Список таблиц

1 Цель работы

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

2 Выполнение лабораторной работы

Создаю каталог для выполнения лабораторной работы №9, перехожу в него и создаю файл lab9-1.asm (рис. 3.9).

```
pazhelobickaya@dk8n68 ~ $ mkdir ~/work/arch-pc/lab09
pazhelobickaya@dk8n68 ~ $ cd ~/work/arch-pc/lab09
pazhelobickaya@dk8n68 ~/work/arch-pc/lab09 $ touch lab9-1.asm
pazhelobickaya@dk8n68 ~/work/arch-pc/lab09 $
```

Рис. 2.1: Создание каталога и файла

Ввожу в файл lab9-1.asm текст программы из листинга 9.1 (рис. 2.2).

```

/afs/.dk.sci.pfu.edu.ru/home/p/a/pazhelobickaya/work/arch-pc/lab09/lab9-1.asm
%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. 2.2: Ввод текста программы

Создаю исполняемый файл, запускаю его и проверяю работу (рис. 2.3).

```

pazhelobickaya@dk8n68 ~/work/arch-pc/lab09 $ nasm -f elf lab9-1.asm
pazhelobickaya@dk8n68 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-1 lab9-1.o
pazhelobickaya@dk8n68 ~/work/arch-pc/lab09 $ ./lab9-1
Введите x: 2
2x+7=11

```

Рис. 2.3: Запуск и проверка файла

Вношу изменения в текст программы (рис. 2.4).


```

GNU nano 6.4 /afs/.dk.sci.pfu.edu.ru/home/p/a/pazhelobickaya/work/arch-pc/lab09/lab9-1.asm
#include 'in_out.asm'
SECTION .data
msg: DB 'Введите x: ',0
result: DB '2(3x-1)+7=',0
SECTION .bss
x: RESB 80
res: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax, msg
call sprint
mov ecx, x
mov edx, 80
call sread
mov eax, x
call atoi
call _calcul ; Вызов подпрограммы _calcul1
mov eax, result
call sprint
mov eax, [res]
call iprintf
call quit
_calcul:
call _subcalcul
mov ebx, 2
mul ebx
add eax, 7
mov [res], eax
ret ; выход из подпрограммы
_subcalcul:
mov ebx, 3
mul ebx
sub eax, 1
ret

```

Рис. 2.4: Изменения текста программы

Создаю исполняемый файл, запускаю его и проверяю работу (рис. 2.5).

```

pazhelobickaya@dk5n52 ~/work/arch-pc/lab09 $ nasm -f elf lab9-1.asm
pazhelobickaya@dk5n52 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-1 lab9-1.o
pazhelobickaya@dk5n52 ~/work/arch-pc/lab09 $ ./lab9-1
Введите x: 4
2(3x-1)+7=29

```

Рис. 2.5: Запуск и проверка файла

Создаю файл lab9-2.asm (рис. 2.6).

```

pazhelobickaya@dk8n68 ~/work/arch-pc/lab09 $ touch lab9-2.asm

```

Рис. 2.6: Создание файла

Ввожу в файл текст программы из листинга 9.2 (рис. 2.7).

```

/afs/.dk.sci.pfu.edu.ru/home/p/a/pazhelobickaya/work/arch-pc/lab09/lab9-2.asm
SECTION .data
msg1: db "Hello, ",0x0
msg1Len: equ $ - msg1
msg2: db "world!",0xa
msg2Len: equ $ - msg2
SECTION .text
global _start
_start:
mov eax, 4
mov ebx, 1
mov ecx, msg1
mov edx, msg1Len
int 0x80
mov eax, 4
mov ebx, 1
mov ecx, msg2
mov edx, msg2Len
int 0x80
mov eax, 1
mov ebx, 0
int 0x80

```

Рис. 2.7: Ввод программы

Создаю исполняемый файл, загружаю его в отладчик gdb. Проверяю работу программы, запуская ее в оболочке GDB с помощью команды run (рис. 2.8).

```

pazhelobickaya@dk5n52 ~/work/arch-pc/lab09 $ nasm -f elf -g -l lab9-2.lst lab9-2.asm
pazhelobickaya@dk5n52 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-2 lab9-2.o
pazhelobickaya@dk5n52 ~/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/p/a/pazhelobickaya/work/arch-pc/lab09/lab9-2
Hello, world!
[Inferior 1 (process 3822) exited normally]
(gdb)

```

Рис. 2.8: Загрузка в отладчик gdb и запуск программы в оболочке GDB

Устанавливаю брейкпоинт на метку _start (рис. 2.9).

```

(gdb) break _start
Breakpoint 1 at 0x8049000: file lab9-2.asm, line 9.
(gdb)

```

Рис. 2.9: Установка брейкпоинта

Запускаю эту программу (рис. 2.10).

```
(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/p/a/pazhelobickaya/work/arch-pc/lab09/lab9-2
Breakpoint 1, _start () at lab9-2.asm:9
9      mov eax, 4
(gdb) █
```

Рис. 2.10: Запуск программы

С помощью команды `disassemble` смотрю дисассимилированный код программы (рис. 2.11).

```
(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.
(gdb) █
```

Рис. 2.11: Просмотр дисассимилированного кода программы

Переключаюсь на отображение команд с Intel'овским синтаксисом, с помощью команды `disassemble` (рис. 2.12).

```

(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.
(gdb) █

```

Рис. 2.12: Переключение на отображение команд с Intel'овским синтаксисом

Основное различие заключается в том, что в режиме Intel пишется сначала сама команда, а потом уже ее машинный код, когда в режиме АТТ идет сначала машинный код и уже потом сама команда.

Включаю режим псевдографики (рис. 2.13).

```

B> 0x08049000 <_start> mov     eax,0x4
    0x08049005 <_start+5> mov     ebx,0x1
    0x0804900a <_start+10> mov     ecx,0x804a000
    0x0804900f <_start+15> mov     edx,0x8
    0x08049014 <_start+20> int     0x80
    0x08049016 <_start+22> mov     eax,0x4
    0x0804901b <_start+27> mov     ebx,0x1
    0x08049020 <_start+32> mov     ecx,0x804a008
    0x08049025 <_start+37> mov     edx,0x7
    0x0804902a <_start+42> int     0x80
    0x0804902c <_start+44> mov     eax,0x1
    0x08049031 <_start+49> mov     ebx,0x0
    0x08049036 <_start+54> int     0x80
native process 4095 In: _start          L9    PC: 0x08049000
(gdb) █

```

Рис. 2.13: Режим псевдографики

Команда layout regs (рис. 2.14).

```
[ 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

native process 4095 In: _start L9 PC: 0x8049000
(gdb) layout regs
(gdb) □
```

Рис. 2.14: Режим псевдографики

Проверяю точку останова с помощью команды info breakpoints (рис. 2.15).

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

Рис. 2.15: Проверка точки останова

Определяю адрес последней инструкции (mov ebx,0x0) и устанавливаю точку останова (рис. 2.16).

```
[ Register Values Unavailable ]

0x804902a <_start+42>    int     0x80
0x804902c <_start+44>    mov     eax,0x1
0x8049031 <_start+49>    mov     ebx,0x0
b+ 0x8049036 <_start+54>    int     0x80
0x8049038                add     BYTE PTR [eax],al
0x804903a                add     BYTE PTR [eax],al
0x804903c                add     BYTE PTR [eax],al

native process 4095 In: _start L9 PC: 0x8049000
(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) break *0x8049036
Breakpoint 2 at 0x8049036: file lab9-2.asm, line 21.
(gdb) □
```

Рис. 2.16: Адрес предпоследней инструкции

Просматриваю информацию о всех установленных точках останова (рис. 2.17).

```
native process 4095 In: _start L9 PC: 0x8049000
(gdb) break *0x8049036
Breakpoint 2 at 0x8049036: file lab9-2.asm, line 21.
(gdb) i b
Num Type Disp Enb Address What
1 breakpoint keep y 0x8049000 lab9-2.asm:9
  breakpoint already hit 1 time
2 breakpoint keep y 0x8049036 lab9-2.asm:21
(gdb)
```

Рис. 2.17: Информация о всех точках останова

Просматриваю содержимое регистров с помощью команды info registers (рис. 2.18).

```
native process 4095 In: _start L9 PC: 0x8049000
eax 0x0 0
ecx 0x0 0
edx 0x0 0
ebx 0x0 0
esp 0xffffc2d0 0xffffc2d0
ebp 0x0 0x0
esi 0x0 0
edi 0x0 0
eip 0x8049000 0x8049000 <_start>
eflags 0x202 [ IF ]
--Type <RET> for more, q to quit, c to continue without paging--
```

Рис. 2.18: Содержимое регистров

Просматриваю значение переменной msg1 по имени (рис. 2.19).

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

Рис. 2.19: Значение переменной msg1 по имени

Просматриваю значение переменной msg2 по адресу (рис. 2.20).

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

Рис. 2.20: Значение переменной msg2 по адресу

Изменяю первый символ переменной msg1 (рис. 2.21).

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

Рис. 2.21: Изменение первого символа переменной msg1

Заменяю первый символ во второй переменной msg2 (рис. 2.22).

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

Рис. 2.22: Замена второго символа в переменной msg2

Вывожу в различных форматах (в шестнадцатеричном, в двоичном форматах и в символьном виде) значение регистра ebx (рис. 2.23).

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

Рис. 2.23: Выведение значений регистра ebx

Изменяю значение регистра ebx с помощью команды set (рис. 2.24).


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

Рис. 2.24: Изменение значений регистра ebx

Завершаю выполнение программы с помощью команды `continue` или `stepi` и выхожу из GDB с помощью команды `quit`.

Копирую файл `lab8-2.asm` в файл `lab9-3.asm` и создаю исполняемый файл (рис. 2.25).

```
pazhelobickaya@dk5n52 ~ $ cp ~/work/arch-pc/lab08/lab8-2.asm ~/work/arch-pc/lab09/lab9-3.asm
pazhelobickaya@dk5n52 ~ $ cd ~/work/arch-pc/lab09
pazhelobickaya@dk5n52 ~/work/arch-pc/lab09 $ nasm -f elf -g -l lab9-3.lst lab9-3.asm
pazhelobickaya@dk5n52 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-3 lab9-3.o
```

Рис. 2.25: Копирование и создание исполняемого файла

Загружаю исполняемый файл в отладчик, указывая аргументы (рис. 2.26).

```

pazhelobickaya@dk5n52 ~/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) 

```

Рис. 2.26: Загрузка исполняемого файла в отладчик

Устанавливаю точку останова перед первой инструкцией в программе и запускаю ее (рис. 2.27).

```

(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab9-3.asm, line 5.
(gdb) run
Starting program: /afs/dk.sci.pfu.edu.ru/home/p/a/pazhelobickaya/work/arch-pc/lab09/lab9-3 аргумент1 аргумент 2 аргумент\ 3

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

```

Рис. 2.27: Запуск программы

Адрес вершины стека храниться в регистре esp и по этому адресу располагается число равное количеству аргументов командной строки (рис. 2.28).

```

(gdb) x/x $esp
0xfffffc290:      0x00000005
(gdb) 

```

Рис. 2.28: Адрес вершины стека

Просматриваю остальные позиции стека (рис. 2.29).

```

(gdb) x/x $esp
0xfffffc290:      0x00000005
(gdb) x/s *(void**)(esp + 4)
0xfffffc527:      "/afs/.dk.sci.pfu.edu.ru/home/p/a/pazhelobickaya/work/arch-pc/lab09/lab9-3"
(gdb) x/s *(void**)(esp + 8)
0xfffffc571:      "аргумент1"
(gdb) x/s *(void**)(esp + 12)
0xfffffc583:      "аргумент"
(gdb) x/s *(void**)(esp + 16)
0xfffffc594:      "2"
(gdb) x/s *(void**)(esp + 20)
0xfffffc596:      "аргумент 3"
(gdb) x/s *(void**)(esp + 24)
0x0:      <error: Cannot access memory at address 0x0>
(gdb) 

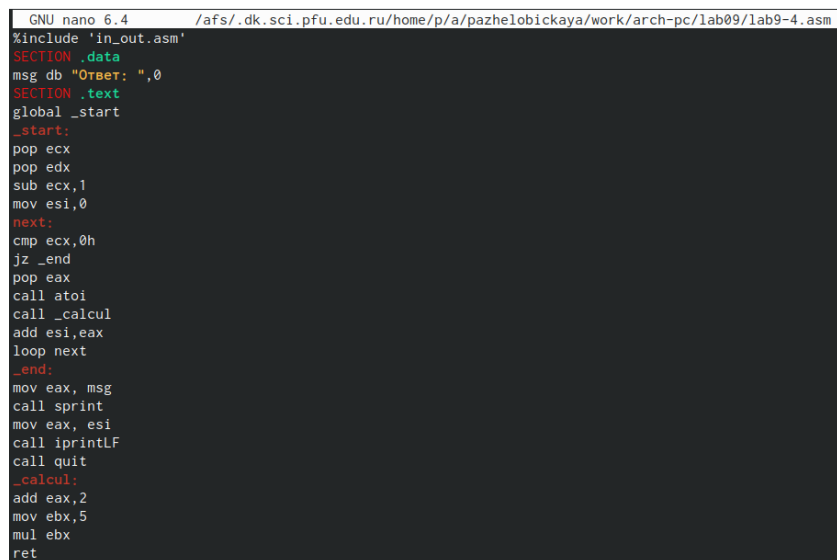
```

Рис. 2.29: Остальные позиции стека

Количество аргументов командной строки 4, значит и шаг равен 4.

3 Выполнение заданий самостоятельной работы

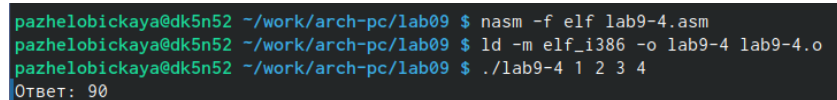
Преобразую программу из лабораторной работы №8, добавляю подпрограмму, которая вычисляет значения функции $f(x)$ (рис. 3.1).



```
GNU nano 6.4 /afs/.dk.sci.pfu.edu.ru/home/p/a/pazhelobickaya/work/arch-pc/lab09/lab9-4.asm
%include 'in_out.asm'
SECTION .data
msg db "Order: ",0
SECTION .text
global _start
_start:
pop ecx
pop edx
sub ecx,1
mov esi,0
next:
cmp ecx,0h
jz _end
pop eax
call atoi
call _calcul
add esi,eax
loop next
_end:
mov eax,msg
call sprint
mov eax,esi
call iprintLF
call quit
_calcul:
add eax,2
mov ebx,5
mul ebx
ret
```

Рис. 3.1: Редактирование файла

Создаю исполняемый файл и ввожу аргументы. Получаю верный ответ (рис. 3.2).



```
pazhelobickaya@dk5n52 ~/work/arch-pc/lab09 $ nasm -f elf lab9-4.asm
pazhelobickaya@dk5n52 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-4 lab9-4.o
pazhelobickaya@dk5n52 ~/work/arch-pc/lab09 $ ./lab9-4 1 2 3 4
Ответ: 90
```

Рис. 3.2: Создание исполняемого файла и проверка его

Создаю файл lab9-5.asm (рис. 3.3).

```
pazhelobickaya@dk5n52 ~/work/arch-pc/lab09 $ touch lab9-5.asm
```

Рис. 3.3: Создание файла

Ввожу в этот файл текст программы из листинга 9.3 (рис. 3.4).

```
GNU nano 6.4 /afs/.dk.sci.pfu.edu.ru/home/p/a/pazhelobickaya/work/arch-pc/lab09/lab9-5.asm
#include 'in_out.asm'
SECTION .data
div: DB 'Результат: ',0
SECTION .text
GLOBAL _start
_start:
; ---- Вычисление выражения (3+2)*4+5
mov ebx,3
mov eax,2
add ebx,eax
mov ecx,4
mul ecx
add ebx,5
mov edi,ebx
; ---- Вывод результата на экран
mov eax,div
call sprint
mov eax,edi
call iprintfLF
call quit
```

Рис. 3.4: Ввод текста программы

Открываю файл в отладчике GDB и запускаю программу (рис. 3.5).

```
pazhelobickaya@dk5n52 ~/work/arch-pc/lab09 $ nasm -f elf -g -l lab9-5.lst lab9-5.asm
pazhelobickaya@dk5n52 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-5 lab9-5.o
pazhelobickaya@dk5n52 ~/work/arch-pc/lab09 $ gdb lab9-5
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 lab9-5...
(gdb)
```

Рис. 3.5: Запуск программы в отладчике

Просматриваю дисассимилированный код программы, ставлю точку останова перед прибавлением 5 и открываю значения регистров на данном этапе (рис. 3.6).

```

(gdb) disassemble _start
Dump of assembler code for function _start:
   0x080490e8 <+0>:    mov     $0x3,%ebx
   0x080490ed <+5>:    mov     $0x2,%eax
   0x080490f2 <+10>:   add     %eax,%ebx
   0x080490f4 <+12>:   mov     $0x4,%ecx
   0x080490f9 <+17>:   mul     %ecx
   0x080490fb <+19>:   add     $0x5,%ebx
   0x080490fe <+22>:   mov     %ebx,%edi
   0x08049100 <+24>:   mov     $0x804a000,%eax
   0x08049105 <+29>:   call    0x804900f <sprint>
   0x0804910a <+34>:   mov     %edi,%eax
   0x0804910c <+36>:   call    0x8049086 <iprintf>
   0x08049111 <+41>:   call    0x80490db <quit>
End of assembler dump.
(gdb) b *0x080490fb
Breakpoint 1 at 0x80490fb: file lab9-5.asm, line 13.
(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/p/a/pazhelobickaya/work/arch-pc/lab09/lab9-5

Breakpoint 1, _start () at lab9-5.asm:13
13      add ebx,5
(gdb) i r
eax            0x8            8
ecx            0x4            4
edx            0x0            0
ebx            0x5            5
esp            0xffffc2d0     0xffffc2d0
ebp            0x0            0
esi            0x0            0
edi            0x0            0
eip            0x80490fb     0x80490fb <_start+19>
eflags         0x202         [ IF ]
cs             0x23          35
ss             0x2b          43
ds             0x2b          43
es             0x2b          43
fs             0x0            0
gs             0x0            0
(gdb)

```

Рис. 3.6: Действия в отладчике

Регистр `ecx` со значением 4 умножается не на `ebx`, сложенным с `eax`, а только с `eax` со значением 2. Следовательно, нужно поменять значения регистров (рис. 3.7).

```

GNU nano 6.4 /afs/.dk.sci.pfu.edu.ru/home/p/a/pazhelobickaya/work/arch-pc/lab09/lab9-5.asm
#include 'in_out.asm'
SECTION .data
div: DB 'Результат: ',0
SECTION .text
GLOBAL _start
_start:
; ---- Вычисление выражения (3+2)*4+5
mov ebx,3
mov eax,2
add eax,ebx
mov ecx,4
mul ecx
add eax,5
mov edi,eax
; ---- Вывод результата на экран
mov eax,div
call sprint
mov eax,edi
call iprintf
call quit

```

Рис. 3.7: Изменения в программе

Запускаю программу и получаю верный ответ (рис. 3.8).

```
pazhelobickaya@dk5n52 ~/work/arch-pc/lab09 $ nasm -f elf -g -l lab9-5.lst lab9-5.asm
pazhelobickaya@dk5n52 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-5 lab9-5.o
pazhelobickaya@dk5n52 ~/work/arch-pc/lab09 $ ./lab9-5
Результат: 25
```

Рис. 3.8: Запуск и проверка программы

Описываются проведённые действия, в качестве иллюстрации даётся ссылка на иллюстрацию (рис. 3.9).

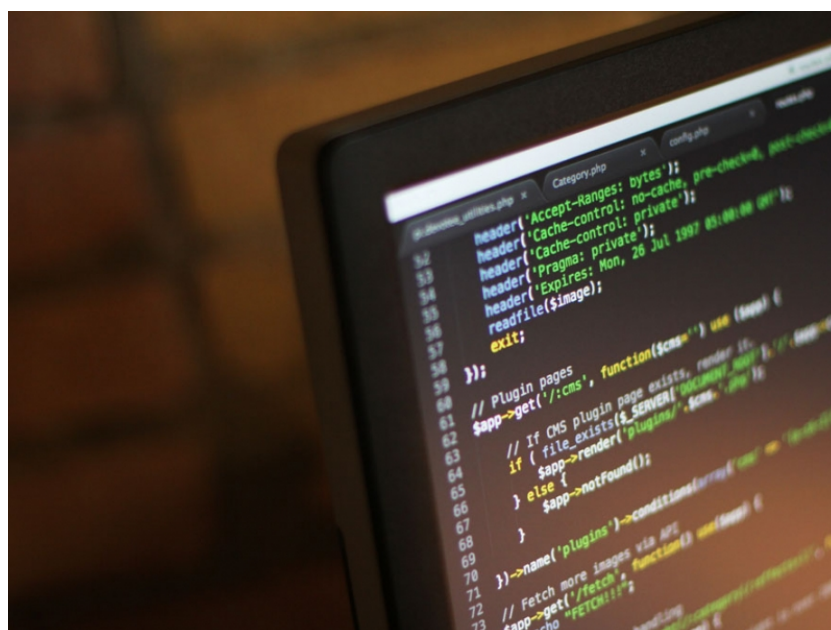


Рис. 3.9: Название рисунка

4 Выводы

я приобрела навыки написания программ с использованием подпрограмм. Ознакомилась с методами отладки при помощи GDB и его основными возможностями.

Список литературы