

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ № 6

дисциплина: *Архитектура компьютера*

Студент: Маслова Анастасия Сергеевна

Группа: НКНбд-01-21

МОСКВА

2021 г.

Цель работы: изучение методов отладки при использовании GDB и его основных возможностей. Получение навыков написания программ с использованием циклов.

Ход работы:

1. Создайте в каталоге *Architecture_PC* (созданном при выполнении Лабораторной работы №1) новый подкаталог с именем *lab06* и в нем файл *lab6-1.asm* (для задания №2) и *lab6-2.asm* (для задания №3).

С помощью уже знакомых команд я создала подкаталог *lab06* и файлы *lab6-1.asm* и *lab6-2.asm* (рис.1).

```
asmaslova@ubuntu2104:~$ ls
Architecture_PC  Documents  Music      Public      Videos
Desktop          Downloads  Pictures   Templates
asmaslova@ubuntu2104:~$ cd Architecture_PC
asmaslova@ubuntu2104:~/Architecture_PC$ mkdir lab06
asmaslova@ubuntu2104:~/Architecture_PC$ cd lab06
asmaslova@ubuntu2104:~/Architecture_PC/lab06$ touch lab6-1.asm lab6-2.asm
```

рис. 1. Создание подкаталога и файлов в нем

2. Напишите программу нахождения произведения чисел от 1 до *N*. *N* передается в программу как аргумент командной строки. Создайте исполняемый файл и проверьте его работу.

С помощью *mcedit* я написала текст программы, а с помощью уже знакомых функций *nasm* и *ld* создала исполняемый файл (рис.2).

```
asmaslova@ubuntu2104:~/Architecture_PC/lab06$ mcedit lab6-1.asm
asmaslova@ubuntu2104:~/Architecture_PC/lab06$ nasm -f elf -g -l lab6-1.lst lab6-1.asm
asmaslova@ubuntu2104:~/Architecture_PC/lab06$ ld -m elf_i386 lab6-1.o lab6-1
```

рис. 2. Создание исполняемого файла

После этого я проверила работу исполняемого файла на числе 10 (рис.3).

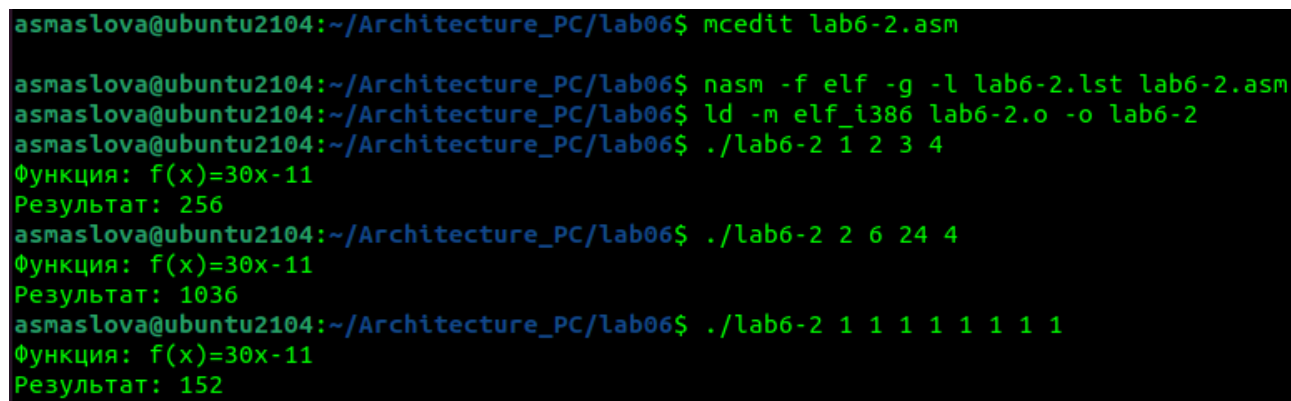
```
asmaslova@ubuntu2104:~/Architecture_PC/lab06$ ./lab6-1
Введите N: 10
Результат: 3628800
```

рис. 3. Проверка работы исполняемого файла

3. Напишите программу, которая находит сумму значений функции $f(x)$ для $x = x_1, x_2, \dots, x_n$, т.е. программа должна выводить значение $f(x_1) + f(x_2) + \dots + f(x_n)$. Значения x_i передаются как аргументы. Вид функции $f(x)$ выбрать из

таблицы 6.1 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 4. Создайте исполняемый файл и проверьте его работу на нескольких наборах $x = x_1, x_2, \dots, x_n$.

С помощью `mcedit` я написала текст программы, после чего создала исполняемый файл и проверила его работу. У меня был 16 вариант, поэтому моей функцией было: $f(x)=30x-11$. (рис.4)



```
asmaslova@ubuntu2104:~/Architecture_PC/lab06$ mcedit lab6-2.asm
asmaslova@ubuntu2104:~/Architecture_PC/lab06$ nasm -f elf -g -l lab6-2.lst lab6-2.asm
asmaslova@ubuntu2104:~/Architecture_PC/lab06$ ld -m elf_i386 lab6-2.o -o lab6-2
asmaslova@ubuntu2104:~/Architecture_PC/lab06$ ./lab6-2 1 2 3 4
Функция: f(x)=30x-11
Результат: 256
asmaslova@ubuntu2104:~/Architecture_PC/lab06$ ./lab6-2 2 6 24 4
Функция: f(x)=30x-11
Результат: 1036
asmaslova@ubuntu2104:~/Architecture_PC/lab06$ ./lab6-2 1 1 1 1 1 1 1 1 1
Функция: f(x)=30x-11
Результат: 152
```

рис. 4. Создание исполняемого файла и проверка его работы.

4. Проведите исследование программы из задания 2 в отладчике `gdb`

- Загрузить программу в отладчик и просмотреть дизассемблированный код программы с помощью команды `disassemble_start`.
- Установить 3 разных точки останова: на второй инструкции указав ее адрес, по имени метки в начале цикла и перед завершением программы (`call quit`) (для удобства можно поставить метку перед инструкцией `call quit`)
- Запустить программу (`run`). После каждой точки останова посмотреть содержимое регистров с помощью команды `info r`.
- Перед завершением программы, после 3 точки останова, выведете в шестнадцатеричном формате, в двоичном формате и в символьном виде (команда `x`) результат произведения в шестнадцатеричном формате и в символьном виде (команда `x`), а также значение регистра `eax` (команда `p`).
- Измените значение произведения с помощью команды `set`.

С помощью команды `gdb` я вызвала отладчик, загрузила в него программу и просмотрела дизассемблированный код программы. (рис. 5).

```

asmaslova@ubuntu2104:~/Architecture_PC/lab06$ gdb
GNU gdb (Ubuntu 10.1-2ubuntu2) 10.1.90.20210411-git
Copyright (C) 2021 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-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
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".
(gdb) file lab6-1
Reading symbols from lab6-1...
(gdb) disassemble _start
Dump of assembler code for function _start:
   0x080490e8 <+0>:    mov     $0x804a000,%eax
   0x080490ed <+5>:    call    0x804900f <sprint>
   0x080490f2 <+10>:   mov     $0x804a028,%ecx
   0x080490f7 <+15>:   mov     $0xa,%edx
   0x080490fc <+20>:   call    0x8049043 <sread>
   0x08049101 <+25>:   mov     $0x804a028,%eax
   0x08049106 <+30>:   call    0x804909c <atoi>
   0x0804910b <+35>:   mov     %eax,0x804a028
   0x08049110 <+40>:   mov     0x804a028,%ecx
   0x08049116 <+46>:   mov     $0x1,%eax
End of assembler dump.

```

рис. 5. Просмотр дизассемблированного кода

После этого я установила 3 точки останова и начала прогонять программу с просмотром регистров на каждой паузе (рис. 6,7,8).

```

(gdb) run
Starting program: /home/asmaslova/Architecture_PC/lab06/lab6-1

Breakpoint 1, _start () at lab6-1.asm:16
16      call sprint
(gdb) info r
eax             0x804a000             134520832
ecx             0x0                  0
edx             0x0                  0
ebx             0x0                  0
esp             0xfffffd130          0xfffffd130
ebp             0x0                  0x0
esi             0x0                  0
edi             0x0                  0
eip             0x80490ed             0x80490ed <_start+5>
eflags         0x202                [ IF ]
cs              0x23                 35
ss              0x2b                 43
ds              0x2b                 43
es              0x2b                 43
fs              0x0                  0
gs              0x0                  0

```

рис. 6. Прогон программы 1

```

(gdb) c
Continuing.
Введите N: 1

Breakpoint 4, _start () at lab6-1.asm:22
22      mov [N],eax
(gdb) info r
eax            0x1            1
ecx            0x0            0
edx            0xa            10
ebx            0x12           18
esp            0xffffd130     0xffffd130
ebp            0x0            0x0
esi            0x0            0
edi            0x0            0
eip            0x804910b       0x804910b <_start+35>
eflags         0x212          [ AF IF ]
cs             0x23           35
ss             0x2b           43
ds             0x2b           43
es             0x2b           43
fs             0x0            0
gs             0x0            0

```

рис. 7. Прогон программы 2

```

(gdb) c
Continuing.

Breakpoint 2, label () at lab6-1.asm:28
28      mul ecx
(gdb) info r
eax            0x1            1
ecx            0x1            1
edx            0xa            10
ebx            0x12           18
esp            0xffffd130     0xffffd130
ebp            0x0            0x0
esi            0x0            0
edi            0x0            0
eip            0x804911b       0x804911b <label>
eflags         0x212          [ AF IF ]
cs             0x23           35
ss             0x2b           43
ds             0x2b           43
es             0x2b           43
fs             0x0            0
gs             0x0            0
(gdb) c
Continuing.
Результат: 1
[Inferior 1 (process 6632) exited normally]
(gdb) █

```

рис. 8. Прогон программы 3

Вывод: в ходе работы я научилась писать программы на ассемблере с циклами, а также на практике познакомилась с принципом работы отладчика GDB.