

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук
Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ № 9

дисциплина: Архитектура компьютера

Студент: Мелкумян Арвин

Группа: НКАбд-04-23

МОСКВА

2023 г.

ПОНЯТИЕ ПОДПРОГРАММЫ. ОТЛАДЧИК GDB

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

Ход работы.

Создадим программу которая вычисляет значение функции $2x+7$ с помощью подпрограммы, где значение x задается с клавиатуры. Исходный код программы показан на рисунке 1.

```
include 'in_out.asm'

SECTION .data
msg: DB 'Введите x: ',0h
result: DB '2x+7=',0h

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

    mov eax, result
    call sprint
    mov eax, [res]
    call iprintlf

    call quit

_calcul:
    mov ebx, 2
    mul ebx
    add eax, 7
    mov [res], eax
    ret
```

Рисунок 1 — Исходный код программы

Результат работы программы показан на рисунке 2.

```
infer@Cameron:~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arh-pc/labs/lab09$ ./lab09-1
Введите x: 5
2x+7=17
infer@Cameron:~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arh-pc/labs/lab09$
```

Рисунок 2 — Результат работы программы

Создадим программу, которая выводит приветствие Hello world! (рисунок 3), скомпилируем ее с флагом -g и откроем в отладчике gdb (рисунок 4).

```
1  SECTION .data
2      msg1: db "Hello, ",0x0
3      msg1Len: equ $ - msg1
4
5      msg2: db "world!",0xa
6      msg2Len: equ $ - msg2
7
8  SECTION .text
9      global _start
10
11 _start:
12     mov eax, 4
13     mov ebx, 1
14     mov ecx, msg1
15     mov edx, msg1Len
16     int 0x80
17
18     mov eax, 4
19     mov ebx, 1
20     mov ecx, msg2
21     mov edx, msg2Len
22     int 0x80
23
24     mov eax, 1
25     mov ebx, 0
26     int 0x80
```

Рисунок 3 — Исходный код программы

```

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from lab09-2...
(gdb) run
Starting program: /home/infer/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arh-pc/labs/lab09/lab09-2
Hello, world!
[Inferior 1 (process 46076) exited normally]
(gdb) break _start
Breakpoint 1 at 0x8049000: file lab09-2.asm, line 12.
(gdb) run
Starting program: /home/infer/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arh-pc/labs/lab09/lab09-2

Breakpoint 1, _start () at lab09-2.asm:12
12      mov     eax, 4
(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) 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)

```

Рисунок 4 — Выполнение программы в отладчике

Как видно из рисунка, в режиме дизассемблирования в режиме intel появляются различия с режимом AT&T. Во-первых в AT&T константы обозначаются символом \$, а имена регистров символом %, во вторых порядок присваивания слева направо.

На рисунке 5 показано пошаговое выполнение программы в отладчике с отображением кода и значений регистров.

```

--Register group: general--
eax      0x8      8      ecx      0x804a000      134520832
edx      0x8      8      ebx      0x1      1
esp      0xffffcef0  0xffffcef0  ebp      0x0      0x0
esi      0x0      0      edi      0x0      0
eip      0x8049016  0x8049016 <_start+22>  eflags   0x202      [ IF ]
cs       0x23      35      ss       0x2b      43
ds       0x2b      43      es       0x2b      43
fs       0x0      0      gs       0x0      0

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,0x804a000
0x8049025 <_start+37>     mov     edx,0x7
0x804902a <_start+42>     int     0x80
0x804902c <_start+44>     mov     eax,0x1
b+ 0x8049031 <_start+49>     mov     ebx,0x0
0x8049036 <_start+54>     int     0x80
0x8049038      add     BYTE PTR [eax],al
0x804903a      add     BYTE PTR [eax],al
0x804903c      add     BYTE PTR [eax],al
0x804903e      add     BYTE PTR [eax],al
0x8049040      add     BYTE PTR [eax],al

native process 46081 In: _start
(gdb) info breakpoints
Num   Type      Disp Enb Address      What
1     breakpoint keep y  0x08049000 lab09-2.asm:12
      breakpoint already hit 1 time
(gdb) break *0x8049031
Breakpoint 2 at 0x8049031: file lab09-2.asm, line 25.
(gdb) info breakpoints
Undefined info command: "braakpoints". Try "help info".
(gdb) info breakpoints
Num   Type      Disp Enb Address      What
1     breakpoint keep y  0x08049000 lab09-2.asm:12
      breakpoint already hit 1 time
2     breakpoint keep y  0x08049031 lab09-2.asm:25
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb)

```

Рисунок 5 — Пошаговое выполнение программы

Переделаем программу вычисления функции из лабораторной работы №8 для работы с подпрограммами. Исходный код полученной программы показан на рисунке 6.

```

1  include 'in_out.asm'
2  SECTION .data
3      msg db "Результат: ",0h
4
5  SECTION .text
6      global _start
7
8  _start:
9      pop ecx
10     pop edx
11     sub ecx, 1
12     mov esi, 0
13
14 next:
15     cmp ecx, 0
16     jz _end
17
18     pop eax
19     call atoi
20
21     call _calc_func
22
23     loop next
24
25 _calc_func:
26     sub eax, 1
27     mov edx, 10
28     mul edx
29     add esi, eax
30     ret
31
32 _end:
33     mov eax, msg
34     call sprint
35     mov eax, esi
36     call iprintLF
37     call quit

```

Рисунок 6 — Исходный код программы

Далее проанализируем код, представленный в методичке листингом 9.3 на предмет ошибок с помощью отладчика. Процесс отладки показан на рисунке 7.

```

Register group: general
eax      0x8      8      ecx      0x4      4      edx      0x0      0
ebx      0x5      5      esp      0xffffcef0  0xffffcef0  ebp      0x0      0x0
esi      0x0      0      edi      0x0      0      eip      0x80490fb  0x80490fb < start+1
eflags   0x206    [ PF IF ]  cs      0x23     35      ss      0x2b     43
ds       0x2b     43      es      0x2b     43      fs      0x0      0
gs       0x0      0

B* 0x80490e8 < start> mov ebx,0x3
0x80490ed < start+5> mov eax,0x2
0x80490f2 < start+10> add ebx,eax
0x80490f4 < start+12> mov ecx,0x4
0x80490f9 < start+17> mul ecx
> 0x80490fb < start+19> add ebx,0x5
0x80490fb < start+22> mov edi,ebx
0x8049100 < start+24> mov eax,0x04a000
0x8049105 < start+29> call 0x804900f <sprint>
0x804910a < start+34> mov eax,edi
0x804910c < start+36> call 0x8049086 <iprintf>
0x8049111 < start+41> call 0x80490db <quit>
0x8049116 add BYTE PTR [eax],al
0x8049118 add BYTE PTR [eax],al
0x804911a add BYTE PTR [eax],al
0x804911c add BYTE PTR [eax],al
0x804911e add BYTE PTR [eax],al
0x8049120 add BYTE PTR [eax],al

native process 46932 In: _start
(gdb) layout regs
(gdb) si
The program is not being run.
(gdb) break *0x80490e8
Breakpoint 1 at 0x80490e8: file lab09-4.asm, line 10.
(gdb) run
Starting program: /home/infer/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arh-pc/labs/lab09/lab09-4

Breakpoint 1, _start () at lab09-4.asm:10
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb)

```

Рисунок 7 — Процесс отладки

По логике работы программы, для получения правильного результата должны сложиться числа 3 и 2, потом сумма умножиться на 4. Из процесса отладки видно, что после 5 итераций в регистре eax оказывается число 8, а не 10, что происходит из-за строки «add ebx, eax», которая должна выглядеть как «add eax, ebx»

Выводы: В ходе лабораторной работы были приобретены навыки использования подпрограмм, а также отладчика gdb.