Понятие подпрограммы. Отладчик GDB.

Лабораторная работа №9

Губайдуллина Софья Романовна

Содержание

1	Цель работы	1
	Задание	
	Теоретическое введение	
	Выполнение лабораторной работы	

1 Цель работы

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

2 Задание

- 1) Реализация подпрограмм в NASM;
- 2) Отладка программам с помощью GDB;
- 3) Работа с данными программы в GDB;
- 4) Обработка аргументов командной строки в GDB;
- 5) Задание для самостоятельной работы.

3 Теоретическое введение

Отладка — это процесс поиска и исправления ошибок в программе. В общем случае его можно разделить на четыре этапа: • обнаружение ошибки; • поиск её местонахождения; • определение причины ошибки; • исправление ошибки. Наиболее популярные способы работы с отладчиком — это использование точек останова и выполнение программы по шагам. Наиболее популярные виды точек останова - точки Breakpoint и Watchpoint.

GDB (GNU Debugger — отладчик проекта GNU) работает на многих UNIX-подобных системах и умеет производить отладку многих языков программирования. Отладчик GDB (как и любой другой отладчик) позволяет увидеть, что происходит «внутри» программы в момент её выполнения или что делает программа в момент сбоя. GDB может выполнять следующие действия: • начать выполнение программы, задав всё,

что может повлиять на её поведение; • остановить программу при указанных условиях; • исследовать, что случилось, когда программа остановилась; • изменить программу так, чтобы можно было поэкспериментировать с устранением эффектов одной ошибки и продолжить выявление других. Синтаксис команды для запуска отладчика имеет следующий вид: gdb [опции] [имя_файла | ID процесса]

Если есть файл с исходным текстом программы, а в исполняемый файл включена информа- ция о номерах строк исходного кода, то программу можно отлаживать, работая в отладчике непосредственно с её исходным текстом. Чтобы программу можно было отлаживать на уровне строк исходного кода, она должна быть откомпилирована с ключом -g. Посмотреть дизассемблированный код программы можно с помощью команды disassemble : (gdb) disassemble _start

Установить точку останова можно командой break (кратко b). Типичный аргумент этой команды — место установки. Его можно задать как имя метки или как адрес. Чтобы не было путаницы с номерами, перед адресом ставится «звёздочка»: (gdb) break * (gdb) b Для продолжения остановленной программы используется команда continue (c) (gdb) c [аргумент]. Команда stepi (кратко sl) позволяет выполнять программу по шагам, т.е. данная команда выполняет ровно одну инструкцию. Команда nexti (или ni) аналогична stepi, но вызов процедуры (функции) трактуется отладчиком как одна инструкция.

Подпрограмма — это, как правило, функционально законченный участок кода, который можно многократно вызывать из разных мест программы. В отличие от простых переходов из подпрограмм существует возврат на команду, следующую за вызовом. Для вызова подпрограммы из основной программы используется инструкция call, которая заносит адрес следующей инструкции в стек и загружает в регистр еір адрес соответствующей подпрограммы, осуществляя таким образом переход. Подпрограмма завершается инструкцией ret, которая извлекает из стека адрес, занесённый туда соответствующей инструкцией call, и заносит его в еір.

4 Выполнение лабораторной работы

1) Начинаю выполнение лабораторной работы с создания нового каталога lab09 и файла lab09-1.asm в нём.(рис. ??).

```
srgubayjdullina@dk3n38 ~ $ mkdir ~/work/arch-pc/lab09
srgubayjdullina@dk3n38 ~ $ ~/work/arch-pc/lab09
bash: /afs/.dk.sci.pfu.edu.ru/home/s/r/srgubayjdullina/work/arch-pc/lab09:
аталог
srgubayjdullina@dk3n38 ~ $ cd ~/work/arch-pc/lab09
srgubayjdullina@dk3n38 ~/work/arch-pc/lab09 $ touch lab09-1.asm
srgubayjdullina@dk3n38 ~/work/arch-pc/lab09 $
```

Создание нового файла и каталога

Ввожу в файл lab09-1.asm текст программы из листинга 9.1. Создаю исполняемый файл и проверьте его работу (рис. ??). Файл работает корректно.

```
srgubayjdullina@dk3n38 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-1 lab09-1.o srgubayjdullina@dk3n38 ~/work/arch-pc/lab09 $ ./lab09-1
Введите х: 4
2x+7=15
```

Проверка работы файла lab09-1.asm

Далее мне необходимо поменять текст программы так, чтобы в нем находилась подпрограмма _subcul для вычисления выражения $\mathbb{Z}(\mathbb{Z}(\mathbb{Z}))$, где \mathbb{Z} вводится с клавиатуры, $\mathbb{Z}(\mathbb{Z}) = 2\mathbb{Z} + 7$, $\mathbb{Z}(\mathbb{Z}) = 3\mathbb{Z} - 1$ (рис. ??)

Измененный файл lab09-1.asm с подпрограммой

Создаю исполняемый файл для измененного lab09-1.asm и проверяю правильность работы программы (рис. ??)

```
srgubayjdullina@dk3n38 ~/work/arch-pc/lab09 $ nasm -f elf lab09-1.asm srgubayjdullina@dk3n38 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-1 lab09-1.o srgubayjdullina@dk3n38 ~/work/arch-pc/lab09 $ ./lab09-1 Bведите х: 4 2x+7=29
```

Работа файла lab09-1.asm с подпрограммой

2) Создаю новый файл lab09-2.asm и ввожу туда листинг 9.2. Проверяю работу (рис. ??)

```
\label{limited_signal_signal} $$ srgubayjdullina@dk3n38 $$ $$ \work/arch-pc/lab09 $ nasm -f elf lab09-2.asm $$ srgubayjdullina@dk3n38 $$ $$ \work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-2 lab09-2.o $$ srgubayjdullina@dk3n38 $$ \work/arch-pc/lab09 $ ./lab09-2 $$ ./lab09-2 $$ Hello, world!
```

Проверка работы файла lab09-2.asm

Для работы с GDB в исполняемый файл необходимо добавить отладочную информацию, для этого транслирую программы с ключом '-g', загружаю исполняемый файл в отладчик gdb (рис. ??) и проверяю работу программы, запустив ее в оболочке GDB с помощью команды run (рис. ??)

```
srgubayjdullina@dk3n38 ~/work/arch-pc/lab09 $ nasm -f elf -g -l lab09-2.lst lab09-2.asm
srgubayjdullina@dk3n38 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-2 lab09-2.o
srgubayjdullina@dk3n38 ~/work/arch-pc/lab09 $ gdb lab09-2
GNU gdb (Gentoo 13.2 vanilla) 13.2
Copyright (C) 2023 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <a href="http://gnu.org/licenses/gpl.html">http://gnu.org/licenses/gpl.html</a>
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 lab09-2...
(gdb)
Загрузка файла в отладчик GDB
```

```
Starting program: /afs/.dk.sci.pfu.edu.ru/home/s/r/srgubayjdullina/work/arch-pc/lab09/lab09-2
Hello, world!
[Inferior 1 (process 3793) exited normally]
```

Запуск при помощи run

Теперь устанавливаю брейкпоинт на метку _start, с которой начинается выполнение любой ассемблерной программы, и запускаю её (рис. ??)

```
(gdb) break _start
Breakpoint 1 at 0x8049000: file lab09-2.asm, line 9.
(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/s/r/srgubayjdullina/work/arch-pc/lab09/lab09-2
Breakpoint 1, _start () at lab09-2.asm:9
       mov eax, 4
```

Установка брейпоинта на метку start

Анализирую дисассимилированный код программы, начиная с метки _start (рис.??), и сравниваю его с выводом отображения команд с Intel'овским синтаксисом (рис. ??).

```
(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
                              mov $0x80
mov $0x4, %eax
mov $0x1, %ebx
mov $0x804a008, %ecx
mov $0x7, %edx
int $0x80
mov $0x1, %eax
   0x08049016 <+22>:
0x0804901b <+27>:
    0x08049020 <+32>:
0x08049025 <+37>:
    0x0804902a <+42>:
    0x0804902c <+44>:
    0x08049031 <+49>:
                                         $0x0,%ebx
    0×08049036 <+54>
                                         $0x80
End of assembler dump.
```

Дисассимилированный код программы

```
(gdb) set disassembly-flavor intel
(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x08049000 <+0>: mov eax,0x4
   0x08049005 <+5>:
   0x08049001 <+10>: mov ecx,0x804a000

0x08049001 <+15>: mov ecx,0x804a000

0x0804901 <+15>: mov edx,0x8

0x0804901 <+20>: int 0x80

0x08049016 <+22>: mov eax,0x4
   0x0804901b <+27>:
0x08049020 <+32>:
                            mov
                                    ebx,0x1
                            mov ecx,0x804a008
   0x08049025 <+37>:
                            mov
                                    edx.0x7
                            int 0x80
   0x0804902a <+42>:
   0x0804902c <+44>:
                                    eax,0x1
                            mov
   0x08049031 <+49>:
                                    ebx,0x0
   0x08049036 <+54>:
End of assembler dump.
```

Отображение команд с Intel-синтаксисом

Безусловно, выводы различны. В режиме ATT имена регистров начинатся с символа %, а в синтаксисе Intel - c \$.

Далее включаю режим псевдографики для работы в нём при помощи layout asm и layout regs. На предыдущих шагах была установлена точка останова по имени метки (_start). Проверяю при помощи команды info breakpoints (i b) (рис. ??).

```
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 <<u>_start+20></u> int
    0x8049016 < start+22> mov
                                eax,0x4
                                                            PC: 0x8049000
native process 4648 In: _start
(gdb) layout regs
(gdb) i b
Num
                  Disp Enb Address
                                      What
      breakpoint keep y 0x08049000 lab09-2.asm:9
       breakpoint already hit 1 time
(gdb)
```

Режим псевдографики

Установливаю еще одну точку останова по адресу инструкции (рис. ??), а затем смотрю информацию обо всех имеющихся точках останова.

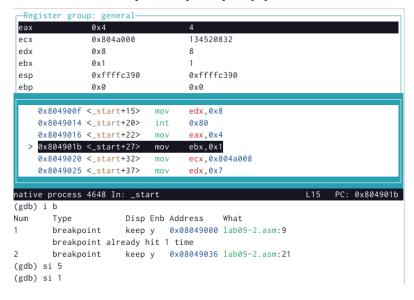
```
(gdb) break *0x08049036
Breakpoint 2 at 0x8049036: file lab09-2.asm, line 21.
(gdb) i b
Num Type Disp Enb Address What
1 breakpoint keep y 0x08049000 lab09-2.asm:9
breakpoint already hit 1 time
2 breakpoint keep y 0x08049036 lab09-2.asm:21
```

Установка точки брейкпоинт по её адресу

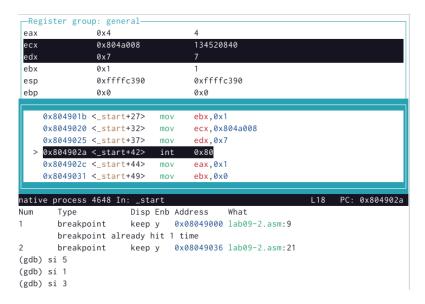
3) Далее по заданию мне необходимо выполнить 5 инструкций с помощью команды stepi (si) и проследить за изменением значений регистров (рис. ??) (рис. ??) (рис. ??) (рис. ??).

```
Register group: general-
eax
               0×8
есх
               0x804a000
                                   134520832
edx
               0x8
                                   8
ebx
               0x1
               0xffffc390
                                  0xffffc390
esp
ebp
               0x0
                                  0x0
    0x804900a <_start+10> mov
                                  ecx,0x804a000
    0x804900f <_start+15> mov
                                  edx,0x8
    0x8049014 <<u>start+20></u>
  > 0x8049016 <_start+22> mov
    0x804901b <_start+27>
    0x8049020 <<u>start+32></u>
                                  ecx,0x804a008
native process 4648 In: _start
                                                          L14 PC: 0x8049016
Breakpoint 2 at 0x8049036: file lab09-2.asm, line 21.
(gdb) i b
Num
      Type
                     Disp Enb Address
                                        What
       breakpoint keep y 0x08049000 lab09-2.asm:9
       breakpoint already hit 1 time
       breakpoint keep y 0x08049036 lab09-2.asm:21
(gdb) si 5
```

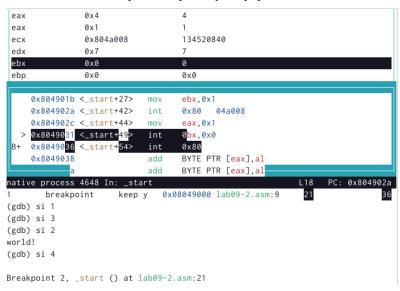
Анализ значений регистров при аргументе si 5



Анализ значений регистров при аргументе si 1



Анализ значений регистров при аргументе si 3



Анализ значений регистров при аргументе si 2 и si 4

Меняются значения регистров eax, ebx, ecx, edx.

Теперь мне необходимо посмотреть значение регистра msg1, а затем значение переменной msg2 (адрес можно узнать по дизассемблированной инструкции) (рис. ??).

```
(gdb) x/1sb &msg1

0x804a000 <msg1>: "Hello, "

(gdb) x/1sb &msg2

0x804a008 <msg2>: "world!\n\034"

(gdb) ■
```

Значение регистра msg1 и переменной msg2

Изменить значение на 'h' для регистра msg1 при помощи set (рис. ??). Так же меняю значение регистра msg2 (рис. ??).

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

Замена значения регистра msg1

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

Замена значения регистра msg2

Необходимо вывести регистр edx в различных форматах (в шестнадцатеричном формате, в двоичном формате и в символьном виде) (рис. ??).

```
(gdb) p/x $edx

$1 = 0x7

(gdb) p/t $edx

$2 = 111

(gdb) p/c $edx

$3 = 7 '\a'
```

Представление регистра edx в различных форматах

Дважды при помощи команды set меняю значение регистра ebx (рис. ??) (рис. ??).

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

Замена значения регистра ebx

Повторная замена значения регистра ebx

Замена произведена по-разному: в первом случае мы меняем символ в строковый вид, а во втором - число не меняется из строкового вида.

Завершаю выполнение программы с помощью команды continue (c) и выхожу (quit).

 Компилирую файл lab8-2.asm, созданный при выполнении лабораторной работы №8, в файл с именем lab09-3.asm, создаю исполняемый файл и при помощи ключа –args загружаю исполняемый файл в отладчик (рис. ??)

```
srgubayjdullina@dk3n38 ~/work/arch-pc/lab09 $ cp ~/work/arch-pc/lab08/lab8-2.asm ~/work/arch-pc/lab09/lab09-3.asm srgubayjdullina@dk3n38 ~/work/arch-pc/lab09 $ nasm -f elf -g -l lab09-3.lst lab09-3.asm srgubayjdullina@dk3n38 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-3 lab09-3. asm srgubayjdullina@dk3n38 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-3 lab09-3. asm srgubayjdullina@dk3n38 ~/work/arch-pc/lab09 $ gbd --args lab09-3 apryweH1 apryweHT 2 'apryweHT 3' GNU gdb (Gentoo 13.2 vanilla) 13.2 Copyright (C) 2023 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <a href="http://gnu.org/licenses/gpl.html">http://gnu.org/licenses/gpl.html</a>
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 configured in "for configuration details.
For bug reporting instructions, please see: <a href="https://buss.gentoo.org/">https://buss.gentoo.org/</a>.
Find the GDB manual and other documentation resources online at: <a href="http://www.gnu.org/software/gdb/documentation/">http://www.gnu.org/software/gdb/documentation/</a>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from lab09-3...
```

Компиляция и копирование lab08-2.asm в новый lab09-3.asm, загрузка в откладчик

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

Установка точки останова

Просматриваю позиции стека (рис. ??).

```
(gdb) x/x $esp
0xffffc340:
              0x00000005
(gdb) x/s *(void**)($esp + 4)
0xffffc5a7
              "/afs/.dk.sci.pfu.edu.ru/home/s/r/srgubayidullina/work/arch-pc/lab09/lab09-3"
(gdb) x/s *(void**)($esp + 8)
              "аргумент1'
0xffffc5f3:
(gdb) x/s *(void**)($esp + 12)
              "аргумент
0xffffc605:
(gdb) x/s *(void**)($esp + 16)
0xffffc616:
(gdb) x/s *(void**)($esp + 20)
0xffffc618:
               "аргумент 3"
(gdb) x/s *(void**)($esp + 24)
       <error: Cannot access memory at address 0x0>
(gdb)
```

Значения позиций стека по адресам

Адреса меняются с шагом 4, т.к. мы имеем 4 аргумента командной строки.

5) Для начала самостоятельной работы я преобразовываю программу из файла lab08-4.asm таким образов, чтобы вычислить значения функции 🛛 (🖺) как подпрограмму. Работаю в новом файле lab09-4.asm:

SECTION .data msg db "Результат", 0 SECTION .text global _start _start: pop ecx pop edx sub ecx,1 mov esi,0 call next next: cmp ecx,0h pop eax call atoi mov edi,3 add eax,10 mul edi add esi,eax jz _end loop next _end: mov eax,msg call sprint mov eax,esi call iprintLF call quit ret

Запускаю программу (рис. ??).

```
srgubayjdullina@dk3n38 ~/work/arch-pc/lab09 $ nasm -f elf lab9-.asm srgubayjdullina@dk3n38 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-4 lab9-4.o srgubayjdullina@dk3n38 ~/work/arch-pc/lab09 $ ./lab9-4 1 3 4 Результат: 102 srgubayjdullina@dk3n38 ~/work/arch-pc/lab09 $
```

Запуск программы файла lab09-4.asm

Приступаю к выполнению второй части самостоятельной работы, а именно: копирую листинг 9.3 вычисления выражения (3 + 2) * 4 + 5 в новый файл lab09-5.asm. При запуске данная программа дает неверный результат (рис. ??). С помощью отладчика GDB, анализируя изменения значений регистров, необходимо найти ошибку и исправьте ее.

```
srgubayjdullina@dk3n38 ~/work/arch-pc/lab09 $ nasm -f elf lab09-5.asm
srgubayjdullina@dk3n38 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-5 lab09-5.o
srgubayjdullina@dk3n38 ~/work/arch-pc/lab09 $ ./lab09-5
Peaynbtat: 10
```

Запуск программы файла lab09-5.asm

Получаю исполняемый файл для GDB и запускаю. Ставлю брейкпоинт на _start, запускаю и открываю режим псевдографики (рис. ??), и теперь ставлю брейкпоинты по адресу на строчки (рис. ??).

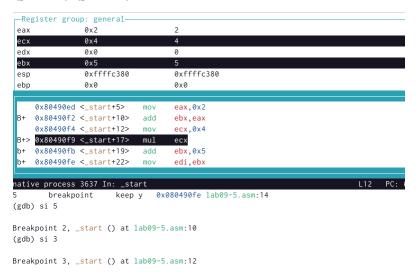
```
srgubayjdullina@dk3n38 ~/work/arch-pc/lab09 $ nasm -f elf lab09-5.asm
srgubayjdullina@dk3n38 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-5 lab09-5.o
srgubayjdullina@dk3n38 ~/work/arch-pc/lab09 $ ./lab09-5
Результат: 10
srgubayjdullina@dk3n38 ~ \begin{tabular}{ll} $ \arrowvert about 1.000 and $ \arrowvert about 1.000 an
srgubayjdullina@dk3n38 ~/work/arch-pc/lab09 $ 1d -m elf_i386 -o lab09-5 lab09-5.o
srgubayjdullina@dk3n38 ~/work/arch-pc/lab09 $ gdb lab09-5
GNU gdb (Gentoo 13.2 vanilla) 13.2
Copyright (C) 2023 Free Software Foundation, Inc.
license GPLv3+: GNU GPL version 3 or later <a href="http://gnu.org/licenses/gpl.html">http://gnu.org/licenses/gpl.html</a>
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 lab09-5...
```

Получение исполняемого файла для запуска GDB



Установка брейкпоинтов на строчки

Теперь при помощи si анализирую изменения значения регистров (рис. ??) (рис. ??) (рис. ??).



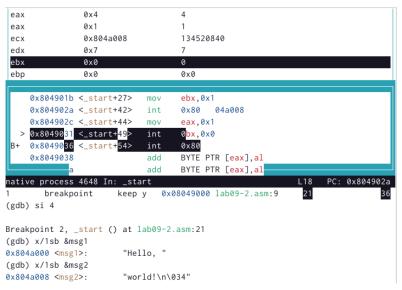
Просмотр значения регистров

```
-Register group: general-
 eax
                0x8
                 0×4
 edx
                0x0
 ebx
                0x5
                0xffffc380
                                     0xffffc380
 esp
                0x0
                                     0x0
 ebp
     0x80490f2 < start+10>
                                     ebx,eax
     0x80490f4 <<u>start+12></u>
                              mov
                                     ecx,0x4
     0x80490f9 < start+17>
 B+> 0x80490fb <_start+19>
     0x80490fe <_start+22>
                                     edi,ebx
     0x8049100 <_start+24>
                                     eax,0x804a000
native process 3637 In: _start
Breakpoint 2, _start () at lab09-5.asm:10
(gdb) si 3
Breakpoint 3, _start () at lab09-5.asm:12
Breakpoint 4, _start () at lab09-5.asm:13
```

Просмотр значения регистров

```
—Register group: general-
 eax
                0x4
есх
                0x0
 edx
 ebx
                                    0xffffc380
 esp
                0xffffc380
                0 x 0
                                    0×0
     0x80490f4 <_start+12>
                                    ecx,0x4
   0x80490f9 <_start+17>
                                    ecx
     0x80490fb <_start+19>
                                    ebx,0x5
    0x80490fe <_start+22>
                                    edi,ebx
     0x8049100 <_start+24> mov
                                    eax,0x804a000
     0x8049105 <_start+29>
                                    0x804900f <sprint>
native process 3637 In: _start
Breakpoint 3, _start () at lab09-5.asm:12
Breakpoint 4, _start () at lab09-5.asm:13
(gdb) si 2
Breakpoint 5, _start () at lab09-5.asm:14
```

Просмотр значения регистров



Просмотр значения регистров

Замечаю некоторый ообенности, а именно: происходит умножение 4 на 2 (есхеах), вместо 4 на 5 (есхеbх), как требуется (регистр ebx показан обнуленным). Функции mov ecx,4; add ebx,eax; mul ecx не связаны друг с другом. Нам необходимо добавить строчку mov edi,ebx. Ещё будет нужно заменить регистр ebx на eax, чтобы связать строчки программы. Теперь программа выглядит таким образом:

%include 'in_out.asm' SECTION .data div: DB 'Peзультат', 0 SECTION .text GLOBAL _start _start: mov ebx,3 mov eax,2 add ebx,eax mov eax,ebx mov ecx,4 mul ecx add eax,5 mov edi,eax mov eax,div call sprint mov eax,edi call iprintLF call quit

Запустим программу для проверки (рис. ??). Программа работает корректно.

```
^[[Asrgubayjdullina@dk3n38 ~/work/arch-pc/lab09 $ nasm -f elf lab09-5.asm srgubayjdullina@dk3n38 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-5 lab09-5.o srgubayjdullina@dk3n38 ~/work/arch-pc/lab09 $ ./lab09-5
```

Выводы

В процессе работы над лабораторной работой я приобрела навыков написания программ с использованием подпрограмм, а так же ознакомилась с методами отладки при помощи GDB и его основными возможностями.