Лабораторная работа № 6

ЧжуЖуйи

10 ноября 2025 г.

Цель работы

Освоение арифметических инструкций языка ассемблера NASM.

Порядок выполнения лабораторной работы

Символьные и численные данные в NASM

1. Создайте каталог для программам лабораторной работы № 6, перейдите в него и создайте файл lab6-1.asm:

```
mkdir ~/work/arch-pc/lab06
cd ~/work/arch-pc/lab06
touch lab6-1.asm
```

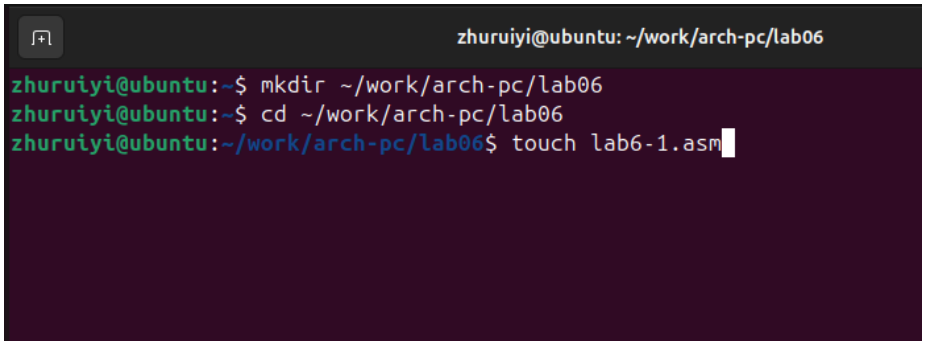

Рис. 1: создать пустой файл

2. Рассмотрим примеры программ вывода символьных и численных значений. Программы будут выводить значения записанные в регистр eax.

Листинг 6.1. Программа вывода значения регистра eax

```
%include 'in_out.asm'

SECTION  .bss
buf1:     RESB 80

SECTION .text
GLOBAL _start
 _start:

 mov eax,'6'
 mov ebx,'4'
 add eax,ebx
 mov [buf1],eax
 mov eax,buf1
 call sprintLF

 call quit
```

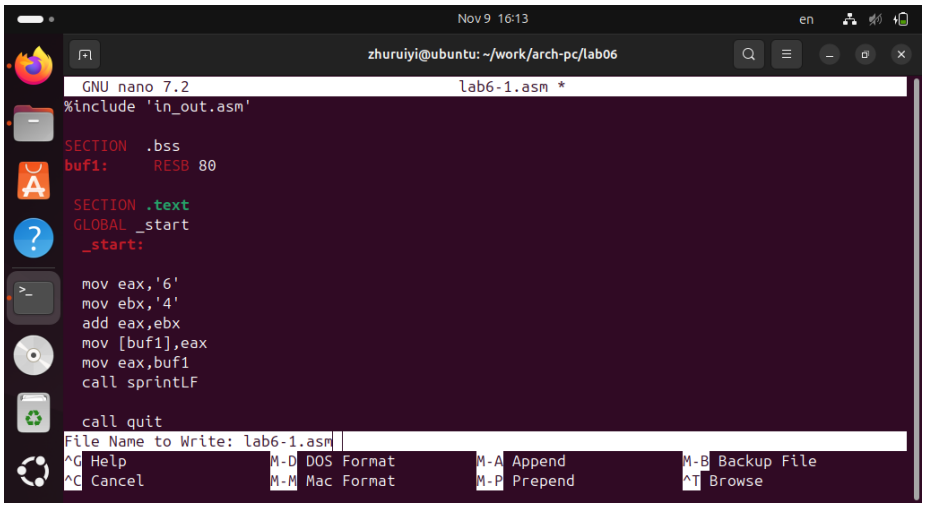

Рис. 2: создадим исполняемый файл

```
nasm -f elf lab6-1.asm
ld -m elf_i386 -o lab6-1 lab6-1.o
./lab6-1
```

```
zhuruiyi@ubuntu:~/work/arch-pc/lab06$ nano lab6-1.asm
zhuruiyi@ubuntu:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
zhuruiyi@ubuntu:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
zhuruiyi@ubuntu:~/work/arch-pc/lab06$ ./lab6-1
j
zhuruiyi@ubuntu:~/work/arch-pc/lab06$
```

Рис. 3: запустим его

3. Далее изменим текст программы и вместо символов, запишем в регистры числа. Исправьте текст программы (Листинг 6.1) следующим образом: замените строки

```
mov eax,'6'
mov ebx,'4'
```

на строки

```
mov eax,6
mov ebx,4
```

```
zhuruiyi@ubuntu: ~/work/arch-pc/lab06
 GNU nano 7.2                    lab6-1.asm *
%include 'in_out.asm'

SECTION  .bss
buf1:     RESB 80

 SECTION .text
 GLOBAL _start
  _start:

  mov eax,6
  mov ebx,4
  add eax,ebx
  mov [buf1],eax
  mov eax,buf1
  call sprintLF

  call quit
```

Рис. 4: заменим исполняемый файл

Рис. 5: запустим его

Даже если программа выполняет сложение чисел (6 + 4 = 10), она всё равно не выведет число 10. Вместо этого она интерпретирует результат (число 10) как символ ASCII. Код ASCII 10 соответствует символу LF (перевод строки). Перевод строки — это управляющий символ, который не виден на экране, но вызывает перемещение курсора на следующую строку. Поэтому вы можете увидеть только пустую строку в выводе или перемещение курсора на новую строку, не видя при этом числа «10».

4. Как отмечалось выше, для работы с числами в файле in_out.asm реализованы подпрограммы для преобразования ASCII символов в числа и обратно. Преобразуем текст программы из Листинга 6.1 с использованием этих функций.

Создайте файл lab6-2.asm в каталоге ~/work/arch-pc/lab06 и введите в него текст программы из листинга 6.2.

```
touch ~/work/arch-pc/lab06/lab6-2.asm
```

Листинг 6.2. Программа вывода значения регистра eax

```
%include 'in_out.asm'

 SECTION .text
 GLOBAL _start
  _start:

  mov eax,'6'
  mov ebx,'4'
  add eax,ebx
  call iprintLF

  call quit
```

Рис. 6: Создадим исполняемый файл и запустим его.

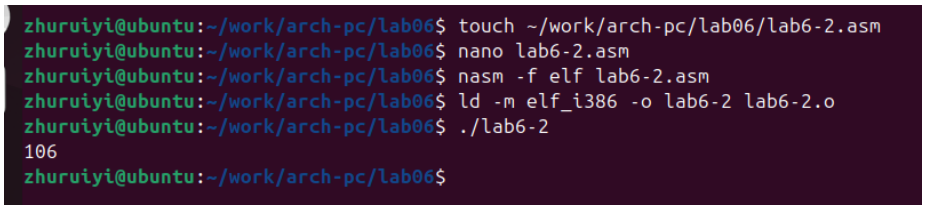

Рис. 7: Создадим исполняемый файл и запустим его.

5. Аналогично предыдущему примеру изменим символы на числа. Замените строки

```
mov eax,'6'
mov ebx,'4'
```

на строки

```
mov eax,6
mov ebx,4
```

Отображаемый результат изменился со 106 на 10.(программа выполняет операцию 6 + 4, и результат в eax равен 10. Функция iprintLF преобразует число 10 в строку «10» и выведет её, добавив символ новой строки.)

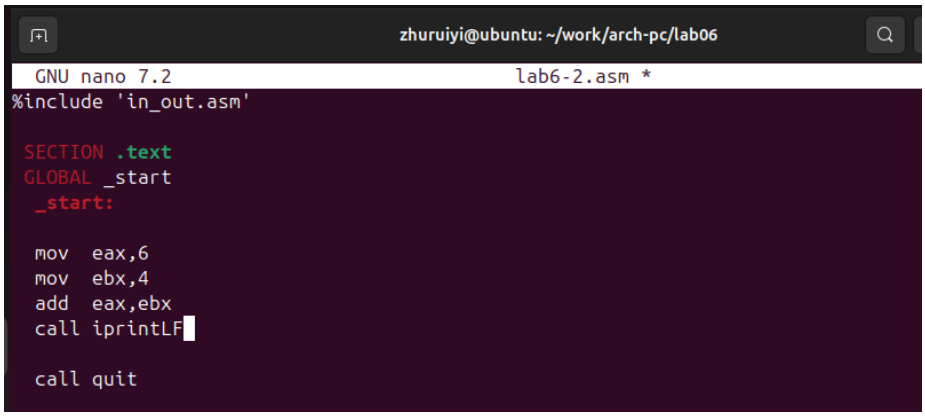

Рис. 8: iprintLF

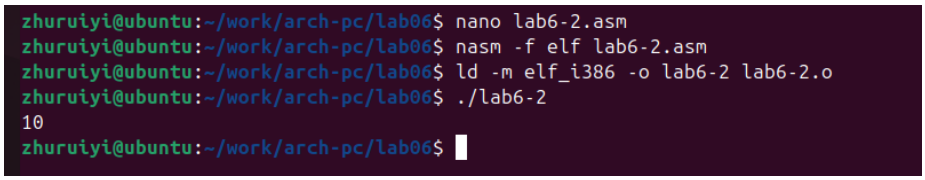

Рис. 9: iprintLF результат

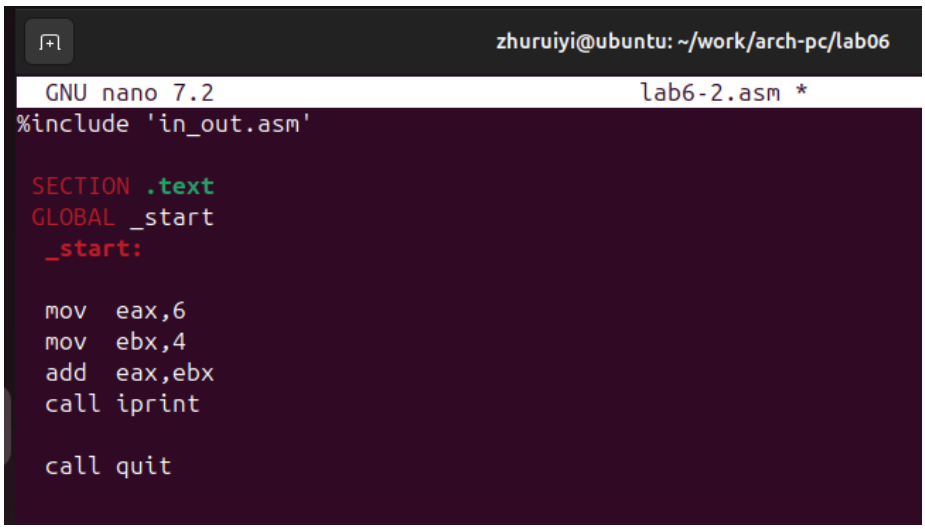

Рис. 10: iprint

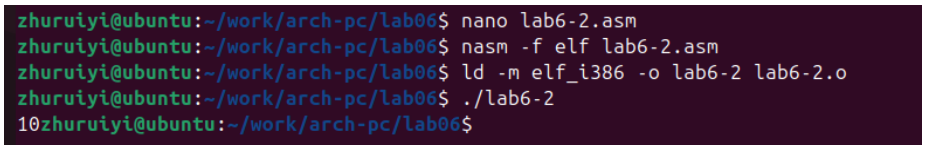

Рис. 11: iprint результат

Различия:

- `iprintLF`: Автоматически добавляет символ новой строки после вывода числовой строки. Курсор перемещается в начало следующей строки после вывода. (рис.8)
- `iprint`: Выводит только числовую строку без добавления символа новой строки. Курсор остаётся в конце вывода. (рис.10)
- Например, если есть дальнейший вывод (без символа новой строки), он будет выведен сразу после числа (как показано на рисунке). (рис.11)

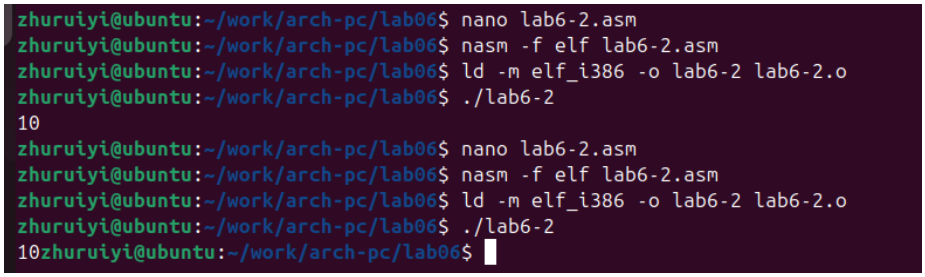

Рис. 12: различие

Выполнение арифметических операций в NASM

6. В качестве примера выполнения арифметических операций в NASM приведем программу вычисления арифметического выражения $f(x) = (5 * 2 + 3)/3$

Создайте файл lab6-3.asm в каталоге ~/work/arch-pc/lab06:

```
touch ~/work/arch-pc/lab06/lab6-3.asm
```

Внимательно изучим текст программы из листинга 6.3 и введём в lab6-3.asm.

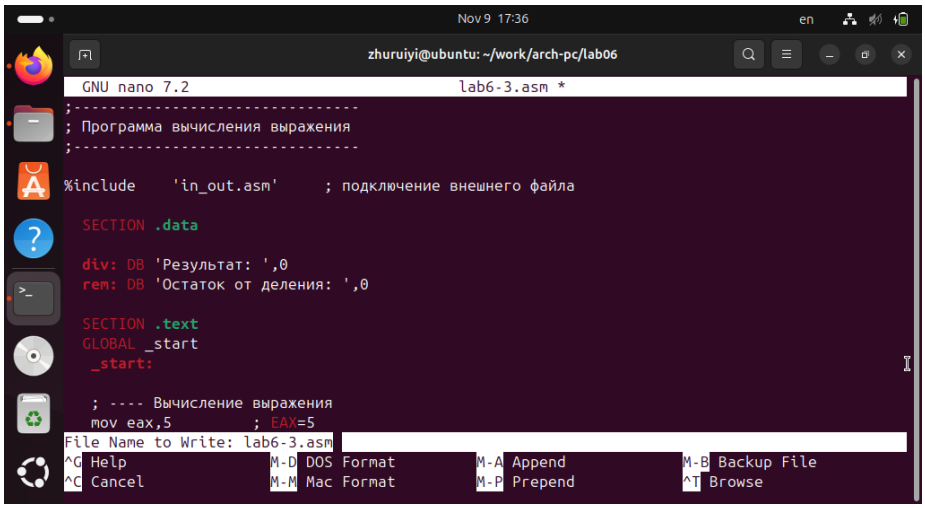

Рис. 13: Листинг 6.3. Программа вычисления выражения $f(x) = (5 * 2 + 3)/3$

```
10zhuruiyi@ubuntu:~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/lab6-3.asm
zhuruiyi@ubuntu:~/work/arch-pc/lab06$ nano lab6-3.asm
zhuruiyi@ubuntu:~/work/arch-pc/lab06$ ./lab6-3
bash: ./lab6-3: No such file or directory
zhuruiyi@ubuntu:~/work/arch-pc/lab06$ nasm -f elf lab6-3.asm
zhuruiyi@ubuntu:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-3 lab6-3.o
zhuruiyi@ubuntu:~/work/arch-pc/lab06$ ./lab6-3
Результат: 4
Остаток от деления: 1
zhuruiyi@ubuntu:~/work/arch-pc/lab06$
```

Рис. 14: Создадим исполняемый файл и запустим его

Измените текст программы для вычисления выражения $f(x) = (4 * 6 + 2)/5$. Создайте исполняемый файл и проверьте его работу.

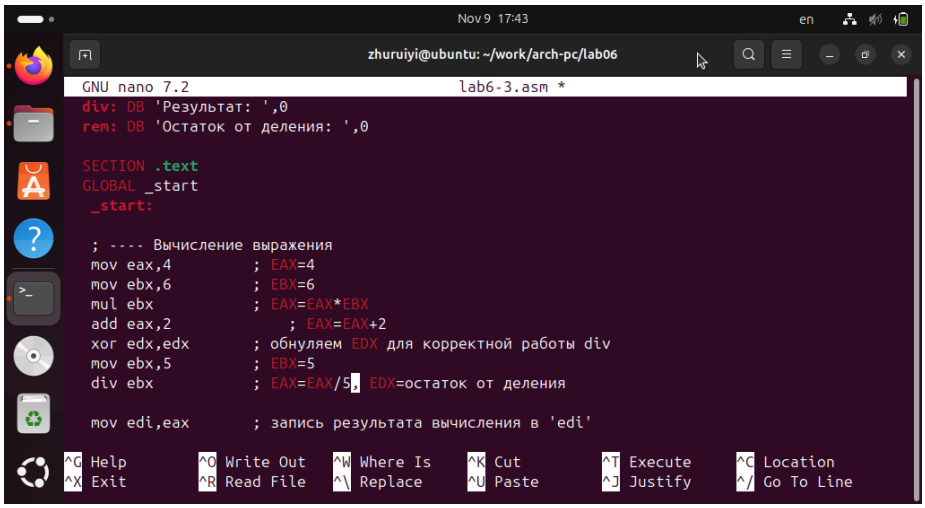

Рис. 15: Создадим исполняемый файл

```
zhuruiyi@ubuntu:~/work/arch-pc/lab06$ nano lab6-3.asm
zhuruiyi@ubuntu:~/work/arch-pc/lab06$ nasm -f elf lab6-3.asm
zhuruiyi@ubuntu:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-3 lab6-3.o
zhuruiyi@ubuntu:~/work/arch-pc/lab06$ ./lab6-3
Результат: 5
Остаток от деления: 1
zhuruiyi@ubuntu:~/work/arch-pc/lab06$
```

Рис. 16: проверим его работу

7. В качестве другого примера рассмотрим программу вычисления варианта задания по номеру студенческого билета, работающую по следующему алгоритму: • вывести запрос на введение № студенческого билета • вычислить номер варианта по формуле: (Sn mod 20) + 1, где Sn – номер студенческого билета (В данном случае a mod b – это остаток от деления a на b). • вывести на экран номер варианта

Внимательно изучите текст программы из листинга 6.4 и введите в файл variant.asm.

Листинг 6.4. Программа вычисления вычисления варианта задания по номеру студенческого билета

```
;-----------------------------
; Программа вычисления варианта
;-----------------------------
```

```
%include    'in_out.asm'
  SECTION .data
  msg: DB 'Введите № студенческого билета: ',0
  rem: DB 'Ваш вариант: ',0

  SECTION .bss
  x:   RESB 80

  SECTION .text
  GLOBAL _start
   _start:

   mov  eax, msg
   call sprintLF

   mov  ecx, x
   mov  edx, 80
   call sread

   mov  eax,x        ; вызов подпрограммы преобразования
   call atoi         ; ASCII кода в число, `eax=x`

   xor edx,edx
   mov ebx,20
   div ebx
   inc edx

   mov  eax,rem
   call sprint
   mov  eax,edx
   call iprintLF

   call quit
```

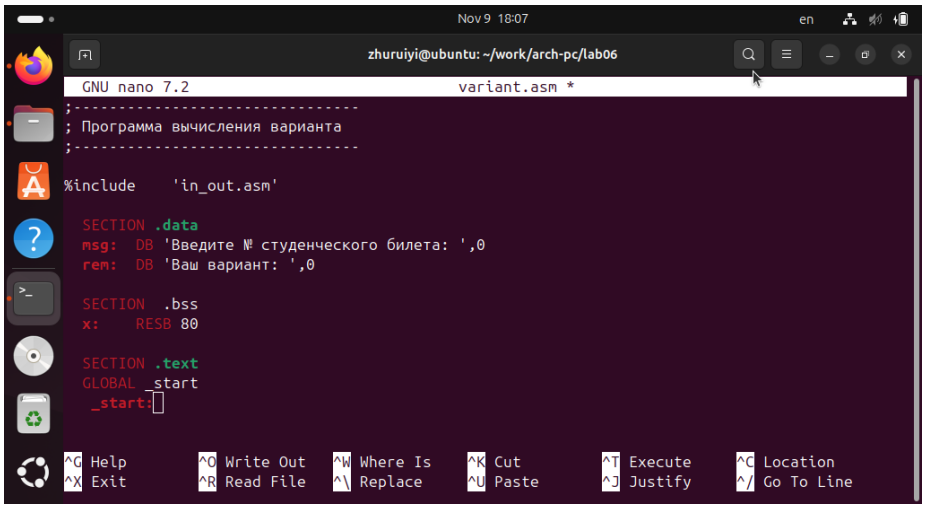

Рис. 17: Создадим исполняемый файл (Листинг 6.4)

```
zhuruiyi@ubuntu:~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/variant.asm
zhuruiyi@ubuntu:~/work/arch-pc/lab06$ nano variant.asm
zhuruiyi@ubuntu:~/work/arch-pc/lab06$ nasm -f elf variant.asm
zhuruiyi@ubuntu:~/work/arch-pc/lab06$ ld -m elf_i386 -o variant variant.o
zhuruiyi@ubuntu:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1032254675
Ваш вариант: 16
zhuruiyi@ubuntu:~/work/arch-pc/lab06$
```

Рис. 18: результат

1. Какие строки листинга 6.4 отвечают за вывод на экран сообщения ‘Ваш вариант:’?

Строки `mov eax,rem` и `call sprint`отвечают за вывод на экран сообщения ‘Ваш вариант:’.

2. Для чего используется следующие инструкции?

```
mov ecx, x
mov edx, 80
call sread
```

Эти инструкции используются для чтения пользовательского ввода с клавиатуры:

- `mov ecx, x`: Сохраняет адрес входного буфера в `ECX`

- `mov edx, 80`: Устанавливает максимальное количество байтов для чтения равным 80
- `call sread`: Вызывает функцию чтения строки.

3. Для чего используется инструкция “call atoi”?

Функция `call atoi` преобразует строку ASCII в целое число. Она преобразует номер строки, хранящийся в `x`, в числовое значение, а результат сохраняется в регистре `EAX`.

4. Какие строки листинга 6.4 отвечают за вычисления варианта?

строки листинга 6.4 отвечают за вычисления варианта:

```
xor edx,edx  ; Очистить EDX
mov ebx,20   ; Установить делитель равным 20
div ebx      ; Выполнить деление, EDX = остаток
inc edx      ; EDX = (Sn mod 20) + 1
```

5. В какой регистр записывается остаток от деления при выполнении инструкции “div ebx”?

Остаток хранится в регистре EDX.

6. Для чего используется инструкция “inc edx”?

Параметр `inc edx` добавляет 1 к значению в EDX, реализуя часть +1 формулы, и преобразует диапазон остатка от 0 до 19 в диапазон номеров вариантов 1-20.

7. Какие строки листинга 6.4 отвечают за вывод на экран результата вычислений?

Строки , выводящие результаты вычислений: `mov eax,edx call iprintFL`.

Задание для самостоятельной работы

1. Написать программу вычисления выражения $y = f(x)$. Программа должна выводить выражение для вычисления, выводить запрос на ввод значения x, вычислять заданное выражение в зависимости от введенного x, выводить результат вычислений. Вид функции $f(x)$ выбрать из таблицы 6.3 вариантов заданий в соответствии с номером полученным при

выполнении лабораторной работы. Создайте исполняемый файл и проверьте его работу для значений $x1$ и $x2$ из 6.3.

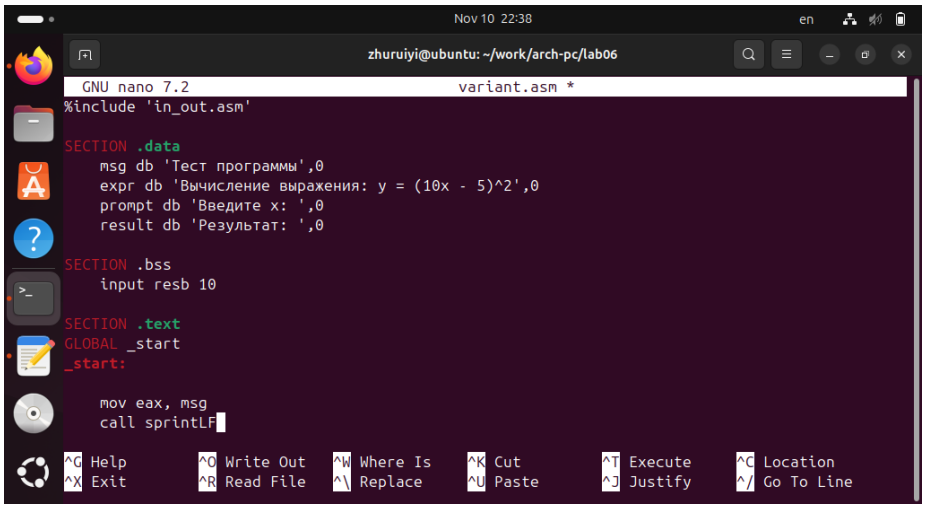

Рис. 19: ответ

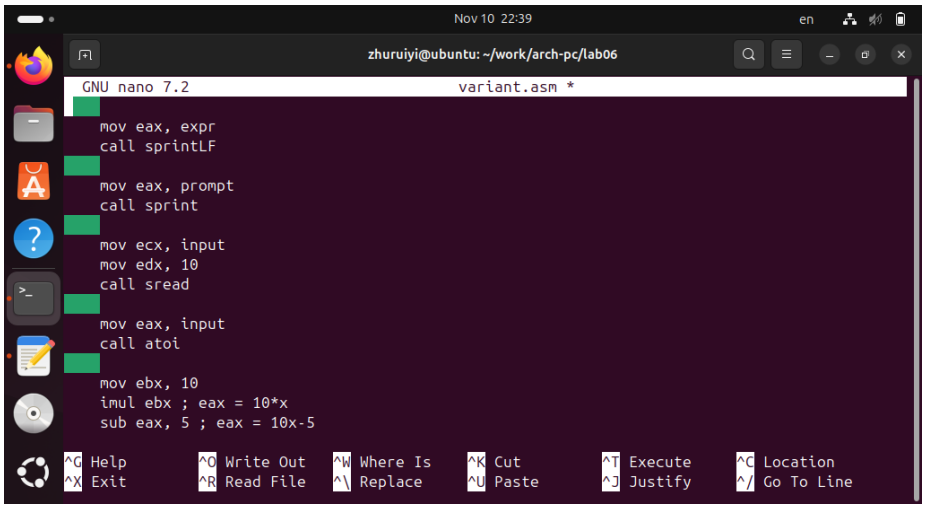

Рис. 20: ответ

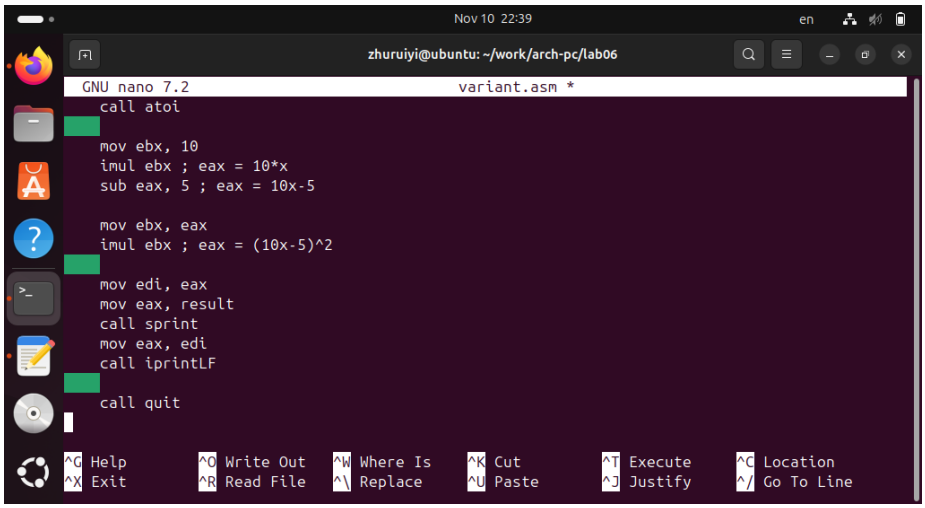

Рис. 21: ответ

```
zhuruiyi@ubuntu:~/work/arch-pc/lab06$ nano variant.asm
zhuruiyi@ubuntu:~/work/arch-pc/lab06$ nasm -f elf variant.asm
zhuruiyi@ubuntu:~/work/arch-pc/lab06$ ld -m elf_i386 -o variant variant.o
zhuruiyi@ubuntu:~/work/arch-pc/lab06$ ./variant
Тест программы
Вычисление выражения: y = (10x - 5)^2
Введите x: 3
Результат: 625
zhuruiyi@ubuntu:~/work/arch-pc/lab06$ ./variant
Тест программы
Вычисление выражения: y = (10x - 5)^2
Введите x: 1
Результат: 25
zhuruiyi@ubuntu:~/work/arch-pc/lab06$
```

Рис. 22: ответ

Вывод:

В ходе изучения лабораторной работы №6 мы освоили базовый синтаксис ассемблера NASM, использование арифметических инструкций, применение библиотек ввода-вывода, поняли кодировку символов и числовое представление, принципы работы регистров, вычисление варианта по номеру студенческого билета (номер билета mod 20) + 1, а также вычисление функций.