Лабораторная работа № 8

ЧжуЖуйи

30 ноября 2025 г.

Цель работы

Приобретение навыков написания программ с использованием циклов и обработкой аргументов командной строки.

Порядок выполнения лабораторной работы

1.Реализация циклов в NASM

Создайте каталог для программам лабораторной работы № 8, перейдите в него и создайте файл lab8-1.asm:

```
mkdir ~/work/arch-pc/lab08
cd ~/work/arch-pc/lab08
touch lab8-1.asm
```

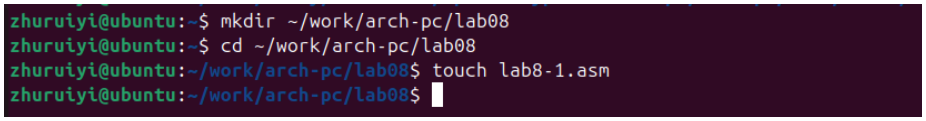

Рис. 1: создадим файл

Листинг 8.1. Программа вывода значений регистра ecx

```
;-------------------------------------------------------------------
; Программа вывода значений регистра 'ecx'
;-------------------------------------------------------------------

%include 'in_out.asm'

SECTION .data
```

```
   msg1 db 'Введите N: ',0h

SECTION .bss
   N:      resb 10

SECTION .text
   global  _start
_start:

; -----   Вывод сообщения 'Введите N: '
   mov  eax,msg1
   call sprint

; -----   Ввод 'N'
   mov  ecx, N
   mov  edx, 10
   call sread

; -----   Преобразование 'N' из символа в число
  mov eax,N
  call atoi
  mov [N],eax

; ------   Организация цикла
  mov ecx,[N]      ; Счетчик цикла, `ecx=N`
label:
  mov [N],ecx
  mov eax,[N]
  call    iprintLF  ; Вывод значения `N`
  loop label        ; `ecx=ecx-1` и если `ecx` не '0'
                                          ; переход на `label`
  call    quit
```

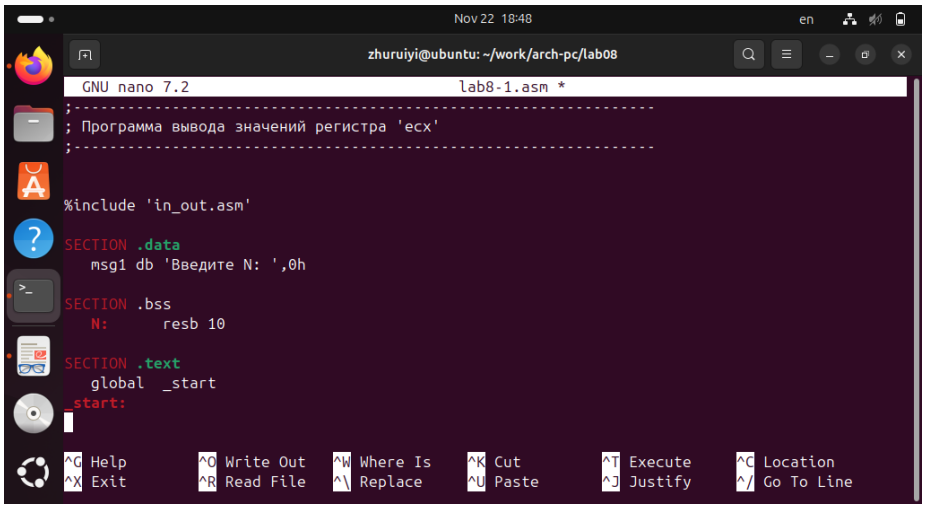

Рис. 2: Введём в файл lab8-1.asm текст программы из листинга 8.1. Создадим исполняемый файл.

```
zhuruiyi@ubuntu:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
zhuruiyi@ubuntu:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-1.o -o lab8-1
zhuruiyi@ubuntu:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 5
5
4
3
2
1
zhuruiyi@ubuntu:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 1
1
zhuruiyi@ubuntu:~/work/arch-pc/lab08$
```

Рис. 3: проверим его работу

Данный пример показывает, что использование регистра ecx в теле цилка loop может привести к некорректной работе программы. Измените текст программы добавив изменение значение регистра ecx в цикле:

```
label:
  sub ecx,1     ; `ecx=ecx-1`
  mov [N],ecx
  mov eax,[N]
  call iprintLF

  loop label
```

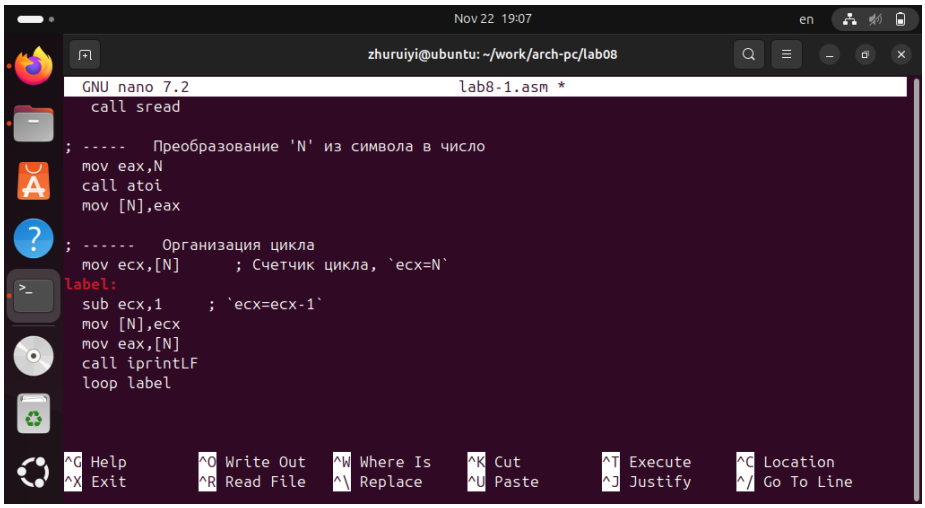

Рис. 4: Создадим исполняемый файл

Рис. 5: проверим его работу

Ответ на вопрос:

(1) В данном цикле регистр ecx принимает значения,которые уменьшаются на 2 при каждой итерации. Это происходит потому, что в каждой итерации цикла: · Инструкция sub ecx,1 явно уменьшает ecx на 1 · Инструкция loop неявно уменьшает ecx на 1 · В результате ecx уменьшается на 2 за каждую итерацию

(2) Нет,число проходов цикла не соответствует значению N, введенному с клавиатуры.Фактическое количество итераций цикла составляет примерно N/2.

Для использования регистра ecx в цикле и сохранения корректности работы программы можно использовать стек. Внесите изменения в текст программы добавив команды push и pop (добавления в стек и извлечения из стека) для сохранения значения счетчика цикла loop:

```
label:
  push ecx              ; добавление значения ecx в стек
  sub ecx,1
  mov [N],ecx
  mov eax,[N]
  call iprintLF
  pop ecx               ; извлечение значения ecx из стека

  loop label
```

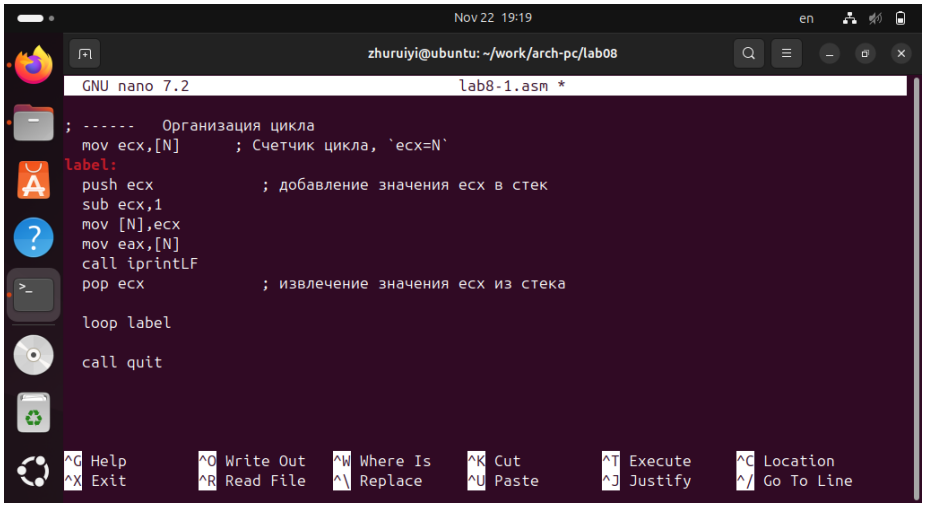

Рис. 6: Создадим исполняемый файл

```
zhuruiyi@ubuntu:~/work/arch-pc/lab08$ nano lab8-1.asm
zhuruiyi@ubuntu:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
zhuruiyi@ubuntu:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-1.o -o lab8-1
zhuruiyi@ubuntu:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 5
4
3
2
1
0
zhuruiyi@ubuntu:~/work/arch-pc/lab08$
```

Рис. 7: проверим его работу

Ответ на вопрос:

(1) Да, после внесения изменений с использованием стека число проходов цикла соответствует значению N, введенному с клавиатуры.

2.Обработка аргументов командной строки

Листинг 8.2. Программа выводящая на экран аргументы командной строки

```
;-------------------------------------------------------------------
; Обработка аргументов командной строки
;-------------------------------------------------------------------
%include    'in_out.asm'

SECTION .text
global  _start

_start:
   pop ecx         ; Извлекаем из стека в `ecx` количество
                   ; аргументов (первое значение в стеке)
   pop edx         ; Извлекаем из стека в `edx` имя программы
                   ; (второе значение в стеке)
   sub ecx, 1      ; Уменьшаем `ecx` на 1 (количество
                   ; аргументов без названия программы)
next:
   cmp ecx, 0      ; проверяем, есть ли еще аргументы
   jz _end         ; если аргументов нет выходим из цикла
                   ; (переход на метку `_end`)
   pop eax         ; иначе извлекаем аргумент из стека
   call sprintLF   ; вызываем функцию печати
   loop next       ; переход к обработке следующего
                   ; аргумента (переход на метку `next`)
_end:
   call quit
```

Создайте файл lab8-2.asm в каталоге ~/work/arch-pc/lab08 и введите в него текст программы из листинга 8.2.

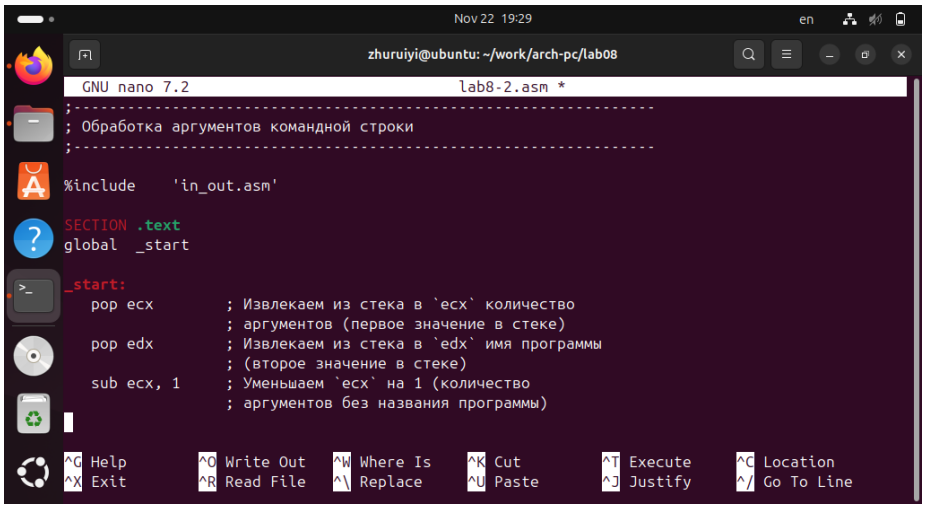

Рис. 8: Создадим файл

Создайте исполняемый файл и запустите его, указав аргументы:

```
./lab8-2 аргумент1 аргумент 2 'аргумент 3'
```

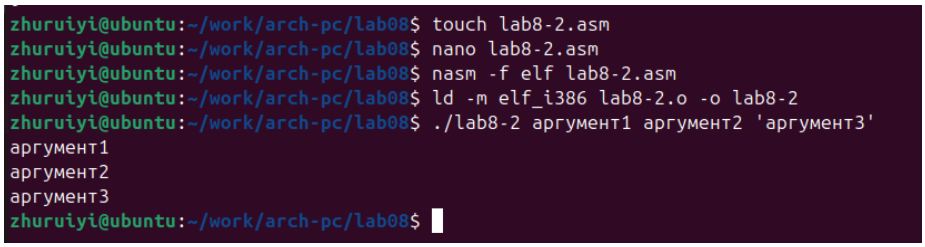

Рис. 9: Результат

Сколько аргументов было обработано программой? Ответ: 3 аргумента.

Листинг 8.3. Программа вычисления суммы аргументов командной строки

```
%include        'in_out.asm'

SECTION .data
msg  db "Результат: ",0

SECTION .text
global  _start
```

```
_start:
   pop ecx     ; Извлекаем из стека в `ecx` количество
               ; аргументов (первое значение в стеке)
   pop edx     ; Извлекаем из стека в `edx` имя программы
               ; (второе значение в стеке)
   sub ecx,1   ; Уменьшаем `ecx` на 1 (количество
               ; аргументов без названия программы)
   mov esi, 0  ; Используем `esi` для хранения
               ; промежуточных сумм
next:
   cmp ecx,0h   ; проверяем, есть ли еще аргументы
   jz _end      ; если аргументов нет выходим из цикла
                ; (переход на метку `_end`)
   pop eax      ; иначе извлекаем следующий аргумент из стека
   call atoi    ; преобразуем символ в число
   add esi,eax  ; добавляем к промежуточной сумме
                ; след. аргумент `esi=esi+eax`
   loop next    ; переход к обработке следующего аргумента

_end:
   mov eax, msg  ; вывод сообщения "Результат: "
   call sprint
   mov eax, esi  ; записываем сумму в регистр `eax`
   call iprintLF ; печать результата
   call quit     ; завершение программы
```

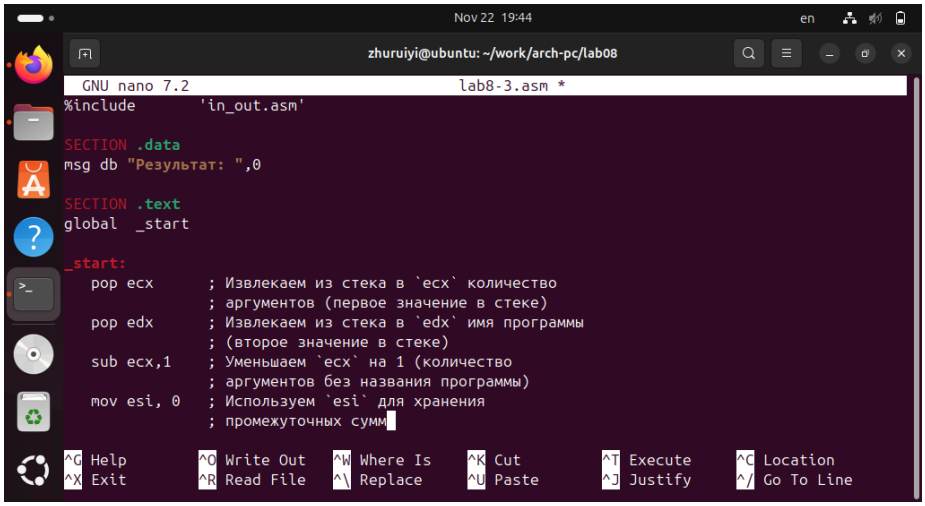

Рис. 10: Создадим файл lab8-3.asm в каталоге ~/work/arch-pc/lab08 и введём в него текст программы из листинга 8.3.

Создайте исполняемый файл и запустите его, указав аргументы. Пример результата работы программы:

```
./lab8-3 12 13 7 10 5
Результат: 47
```

```
zhuruiyi@ubuntu:~/work/arch-pc/lab08$ touch lab8-3.asm
zhuruiyi@ubuntu:~/work/arch-pc/lab08$ nano lab8-3.asm
zhuruiyi@ubuntu:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
zhuruiyi@ubuntu:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-3.o -o lab8-3
zhuruiyi@ubuntu:~/work/arch-pc/lab08$ ./lab8-3 12 13 7 10 5
Результат: 47
zhuruiyi@ubuntu:~/work/arch-pc/lab08$
```

Рис. 11: Результат

Измените текст программы из листинга 8.3 для вычисления произведения аргументов командной строки.

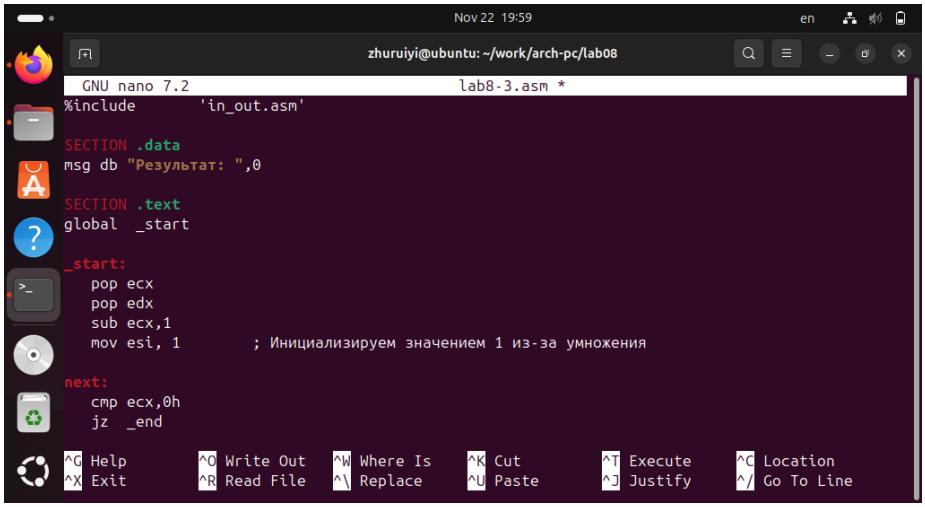

Рис. 12: Изменим текст программы

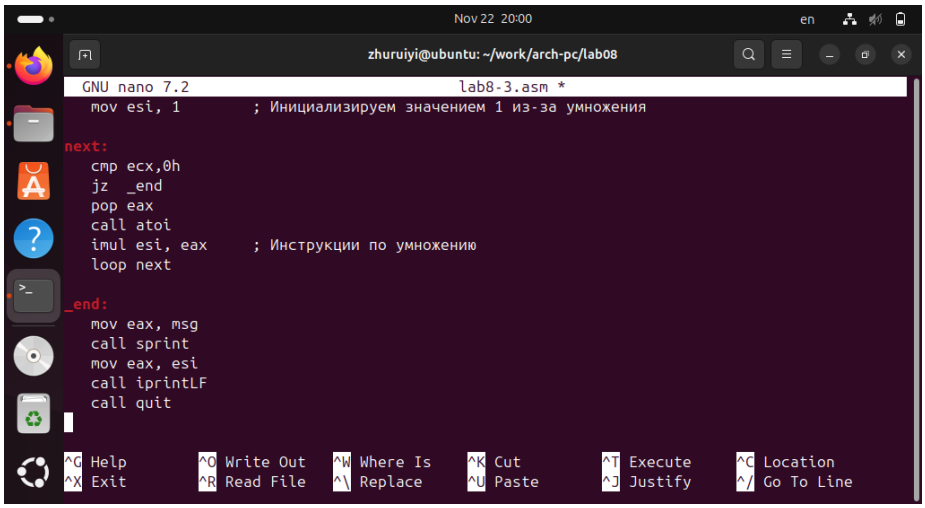

Рис. 13: Изменим текст программы

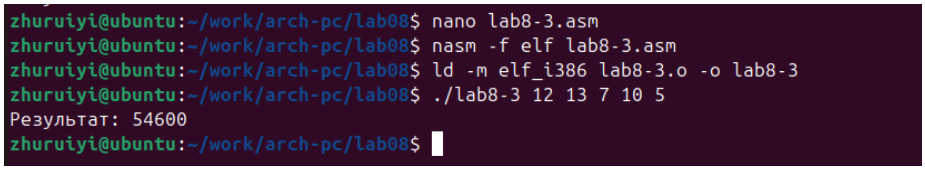

Рис. 14: Результат

Задание для самостоятельной работы

1. Напишите программу, которая находит сумму значений функции $f(x)$ для $x = x1, x2, ..., xn$, т.е. программа должна выводить значение $f(x1) + f(x2) + ... + f(xn)$. Значения xi передаются как аргументы. Вид функции $f(x)$ выбрать из таблицы 8.1 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу на нескольких наборах $x = x1, x2, ..., xn$.

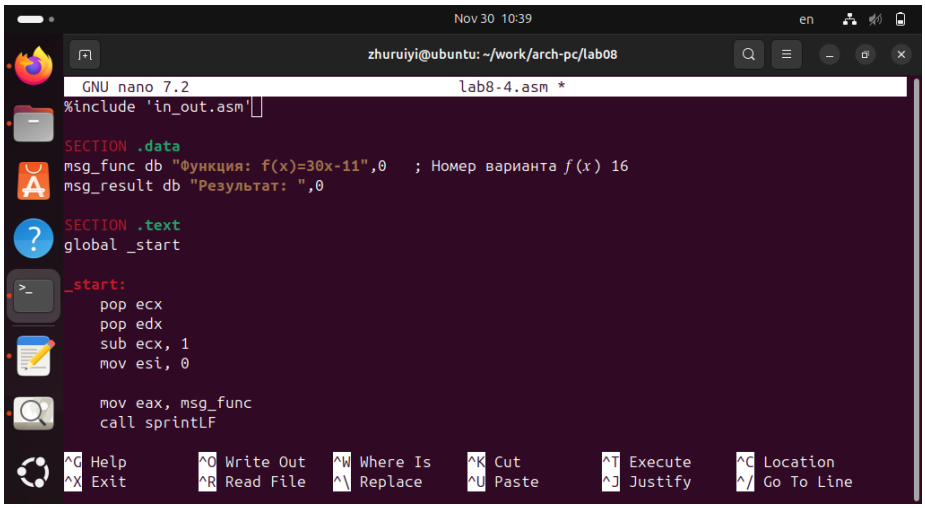

Рис. 15: Создадим исполняемый файл

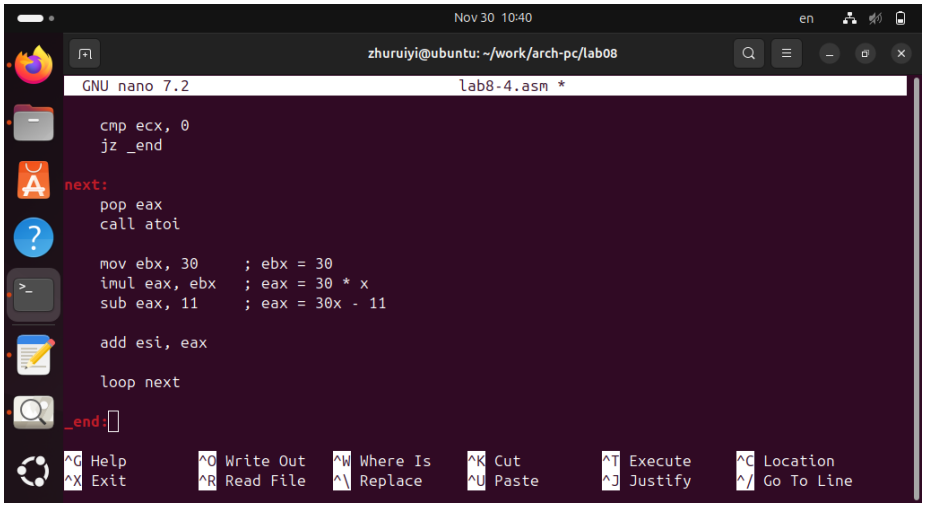

Рис. 16: Создадим исполняемый файл

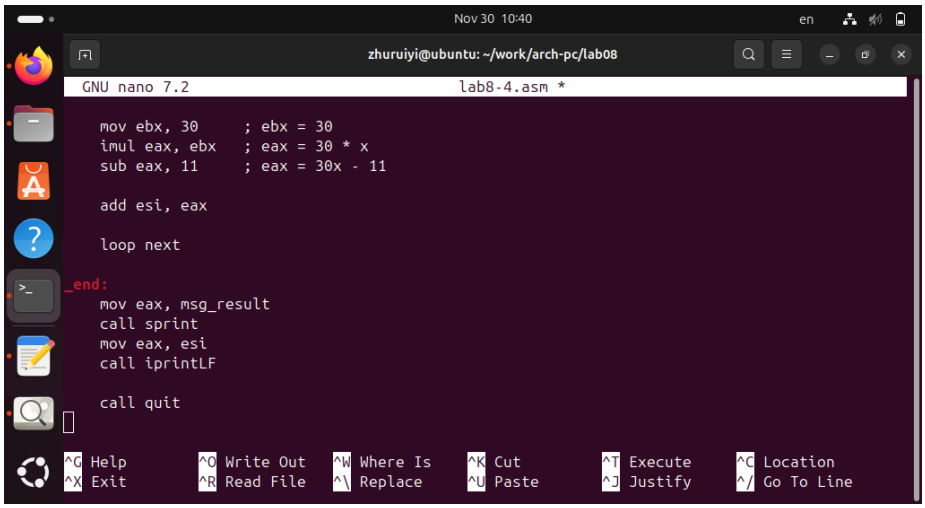

Рис. 17: Создадим исполняемый файл

Работы программы для функции $f(x) = 30x - 11$ и набора $x1 = 1$, $x2 = 7$, $x3 = 5$, $x4 = 7$:

```
./lab8-4 1 7 5 7
Функция: f(x)=30x-11
Результат: 556
```

```
zhuruiyi@ubuntu:~/work/arch-pc/lab08$ touch lab8-4.asm
zhuruiyi@ubuntu:~/work/arch-pc/lab08$ nano lab8-4.asm
zhuruiyi@ubuntu:~/work/arch-pc/lab08$ nasm -f elf lab8-4.asm
zhuruiyi@ubuntu:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-4.o -o lab8-4
zhuruiyi@ubuntu:~/work/arch-pc/lab08$ ./lab8-4 1 7 5 7
Функция: f(x)=30x-11
Результат: 556
zhuruiyi@ubuntu:~/work/arch-pc/lab08$
```

Рис. 18: Результат

Вывод:

В ходе выполнения лабораторной работы №8 мы глубоко изучили правильные методы управления циклами, работы со стеком и обработки аргументов в ассемблере, в частности, научились избегать распространенных проблем конфликтов регистров.