ОТЧЕТ О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ №7

дисциплина: Архитектура компьютера

Шилоносов Данил Вячеславович

Содержание

Список иллюстраций

1 Цель работы

Освоение арифметических инструкций языка ассемблера NASM.

2 Выполнение лабораторной работы

Создадим каталог для программ лабораторной работы №7, перейдем в нее и создадим файл lab7-1.asm. (рис. 2.1)

```
dvshilonosov@dvshilonosov-RUDN:~$ mkdir ~/work/arch-pc/lab07
dvshilonosov@dvshilonosov-RUDN:~$ cd ~/work/arch-pc/lab07
dvshilonosov@dvshilonosov-RUDN:~/work/arch-pc/lab07$ touch lab7-1.asm
```

Рис. 2.1: Создание каталога лабораторной программы и первого текстового файла языка ассемблера

Рассмотрим примеры программ вывода символьных и численных значений. Введем в файл lab7-1.asm текст программы из листинга 7.1. (рис. 2.2)

```
/home/dvshilonosov/work/arch-pc/lab07/lab7-1.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. 2.2: Текст программы из листинга 7.1. в файле lab7-1.asm

Для корректной работы программы подключаемый файл in_out.asm должен лежать в том же каталоге, что и файл с текстом программы. Перед созданием исполняемого файла создадим копию файла in_out.asm в каталоге ~/work/arch-pc/lab07. (рис. 2.3, 2.4)

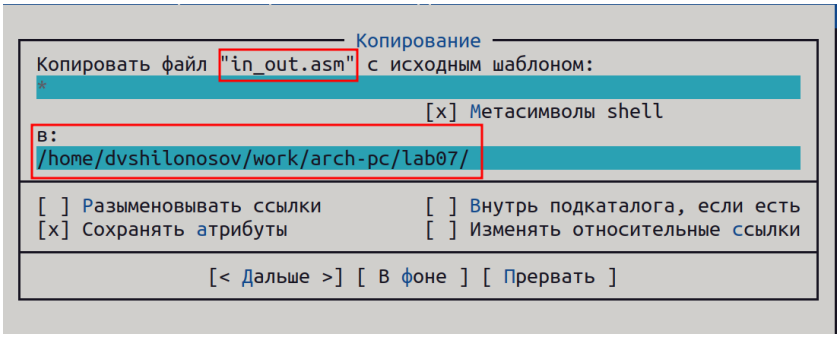

Рис. 2.3: Копирование файла in_out.asm

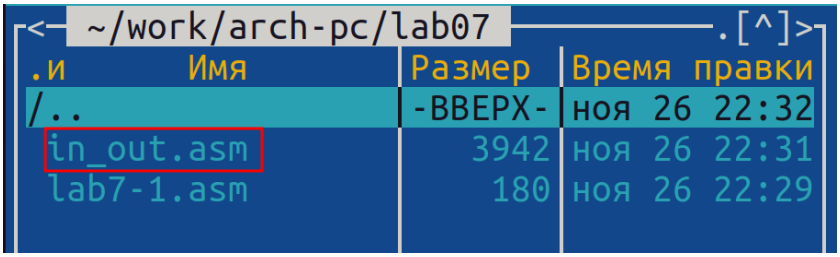

Рис. 2.4: Файл in_out.asm в директории ~/work/arch-pc/lab07

В данной программе в регистр eax записывается символ 6 (mov eax, ‘6’), в регистр ebx символ 4 (mov ebx, ‘4’). Далее к значению в регистре eax прибавляем значение регистра ebx (add eax, ebx, результат сложения запишется в регистр eax). Далее выводим результат. Так как для работы функции sprintLF в регистр eax должен быть записан адрес, необходимо использовать дополнительную переменную. Для этого запишем значение регистра eax в переменную buf1 (mov [buf1], eax), а затем запишем адрес переменной buf1 в регистр eax (mov eax, buf1) и вызовем функцию sprintLF.

Создадим исполняемый файл текста lab7-1.asm и запустим его. (рис. 2.5)

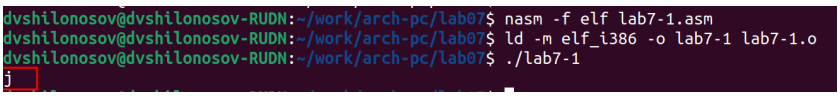

Рис. 2.5: Создание и запуск исполняемого файла текста lab7-1.asm

В данном случае при выводе значения регистра eax мы ожидаем увидеть число 10. Однако результатом будет символ j. Это происходит потому, что код символа 6 равен 00110110 в двоичном представлении (или 54 в десятичном представлении), а код символа 4 – 00110100 (52). Команда add eax, ebx запишет в регистр eax сумму кодов – 01101010 (106), что в свою очередь является кодом символа j (см. таблицу ASCII в приложении).

Далее изменим текст программы и вместо символов запишем в регистры числа. (рис. 2.6)

```
/home/dvshilonosov/work/arch-pc/lab07/lab7-1.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. 2.6: Редактирование файла lab7-1.asm: замена символов на числа

Создадим исполняемый файл и запустим его. (рис. 2.7)

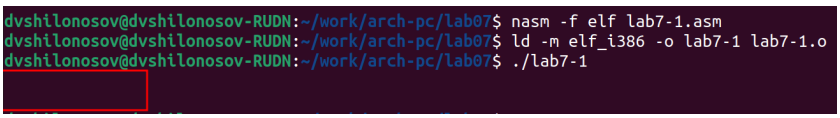

Рис. 2.7: Создание и запуск исполняемого файла отредактированного текста lab7-1.asm

Как и в предыдущем случае при исполнении программы мы не получим число 10. Пользуясь таблицей ASCII, определим, что символу ‘8’ соответствует код 10 (nl). Этот символ не отображается на экране, потому что он соответствует переходу курсора на новую строку (nl - “next line”).

Для работы с числами в файле in_out.asm реализованы подпрограммы для преобразования ASCII-символов в числа и обратно. Преобразуем текст программы из Листинга 7.1 с использованием этих функций.

Создадим файл lab7-2.asm в каталоге ~/work/arch-pc/lab07 и введем в него текст программы из листинга 7.2. (рис. 2.8, 2.9)

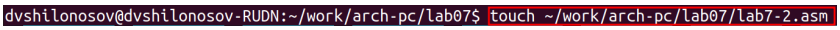

Рис. 2.8: Создание файла lab7-2.asm

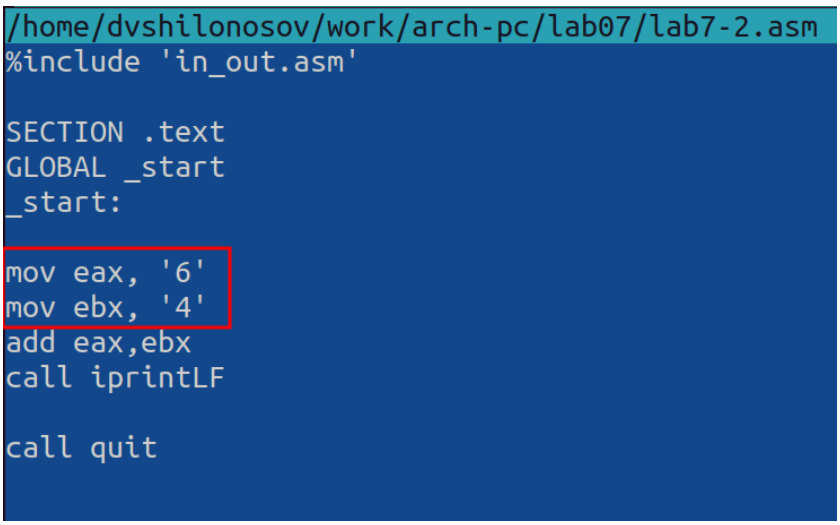

Рис. 2.9: Текст программы из листинга 7.2. в файле lab7-2.asm

Создадим исполняемый файл и запустим его. (рис. 2.10)

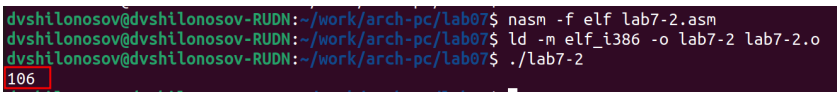

Рис. 2.10: Создание и запуск исполняемого файла текста lab7-2.asm

В результате работы программы мы получим число 106. В данном случае, как и в первом, команда add складывает коды символов ‘6’ и ‘4’ (54+52=106). Однако, в отличие от программы из листинга 7.1, функция iprintLF позволяет вывести число, а не символ, кодом которого является это число.

Аналогично предыдущему примеру изменим символы на числа. (рис. 2.11)

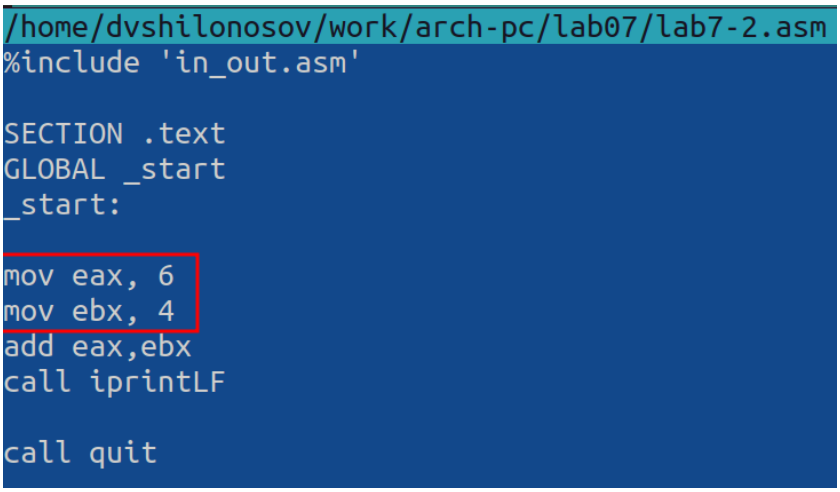

Рис. 2.11: Отредактированный текст файла lab7-2.asm

Создадим исполняемый файл и запустим его. (рис. 2.12)

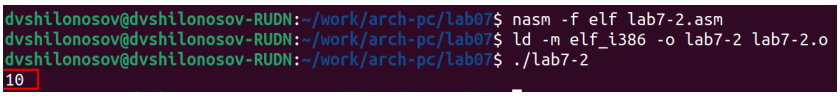

Рис. 2.12: Создание и запуск исполняемого файла отредактированного текста lab7-2.asm

Результат: на экране выведено число 10.

Заменим функцию iprintLF на iprint. (рис. 2.13)

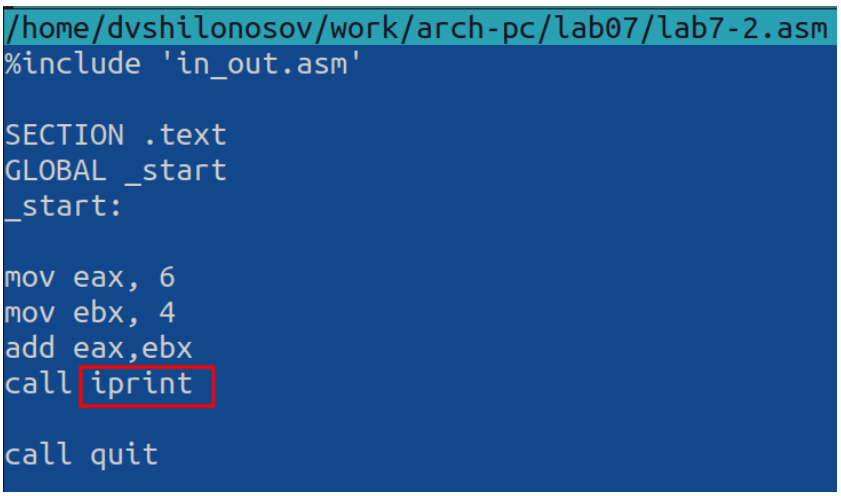

Рис. 2.13: Замена функции iprintLF на iprint в тексте файла lab7-2.asm

Создадим исполняемый файл текста lab7-2.asm и запустим его. (рис. 2.14)

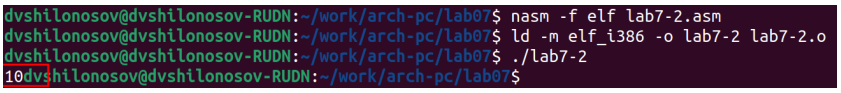

Рис. 2.14: Создание и запуск исполняемого файла отредактированного текста lab7-2.asm

Результат: на экране выведено число 10 без символа перехода на новую строку.

Вывод функций iprintLF и iprint отличается тем, что первая из них, в отличие от второй, после вывода содержимого на экран переводит курсор на новую строку.

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

Создадим файл lab7-3.asm в каталоге ~/work/arch-pc/lab07. (рис. 2.15)

Рис. 2.15: Создание файла lab7-3.asm

Внимательно изучим текст программы из листинга 7.3 и введем в lab7-3.asm. (рис. 2.16)

```
/home/dvshilonosov/work/arch-pc/lab07/lab7-3.asm
;-------------------------------
; Программа вычисления выражения
;-------------------------------

%include 'in_out.asm' ; подключение внешнего файла

SECTION .data

div: DB 'Результат: ', 0
rem: DB 'Остаток от деления: ', 0

SECTION .text
GLOBAL _start
_start:

; ---- Вычисление выражения
mov eax,5                       ; EAX=5
mov ebx,2                       ; EBX=2
mul ebx                         ; EAX=EAX*EBX
add eax,3                       ; EAX=EAX+3
xor edx,edx                     ; обнуляем EDX для корректной работы div
mov ebx,3                       ; EBX=3
div ebx                         ; EAX=EAX/3, EDX=остаток от деления

mov edi,eax                     ; запись результата вычисления в 'edi'

; ---- Вывод результата на экран

mov eax,div                     ; вызов подпрограммы печати
call sprint                     ; сообщения 'Результат: '
mov eax,edi                     ; вызов подпрограммы печати значения
call iprintLF                   ; из 'edi' в виде символов

mov eax, rem                    ; вызов подпрограммы печати
call sprint                     ; сообщения 'Остаток от деления: '
mov eax,edx                     ; вызов подпрограммы печати значения
call iprintLF                   ; из 'edx' (остаток) в виде символов

call quit                       ; вызов подпрограммы завершения
```

Рис. 2.16: Текст программы из листинга 7.3. в файле lab7-3.asm

Создадим исполняемый файл и запустим его. (рис. 2.17)

```
dvshilonosov@dvshilonosov-RUDN:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
dvshilonosov@dvshilonosov-RUDN:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-3 lab7-3.o
dvshilonosov@dvshilonosov-RUDN:~/work/arch-pc/lab07$ ./lab7-3
Результат: 4
Остаток от деления: 1
```

Рис. 2.17: Создание и запуск исполняемого файла текста lab7-3.asm

Изменим текст программы для вычисления выражения f(x) = (4 * 6 + 2) / 5. Создадим исполняемый файл и проверим его работу. (рис. 2.18, 2.19)

```
/home/dvshilonosov/work/arch-pc/lab07/lab7-3.asm
;-------------------------------
; Программа вычисления выражения
;-------------------------------

%include 'in_out.asm' ; подключение внешнего файла

SECTION .data

div: DB 'Результат: ', 0
rem: DB 'Остаток от деления: ', 0

SECTION .text
GLOBAL _start
_start:

; ---- Вычисление выражения
mov eax,4                       ; EAX=5
mov ebx,6                       ; EBX=2
mul ebx                         ; EAX=EAX*EBX
add eax,2                       ; EAX=EAX+3
xor edx,edx                     ; обнуляем EDX для корректной работы div
mov ebx,5                       ; EBX=3
div ebx                         ; EAX=EAX/3, EDX=остаток от деления

mov edi,eax                     ; запись результата вычисления в 'edi'

; ---- Вывод результата на экран

mov eax,div                     ; вызов подпрограммы печати
call sprint                     ; сообщения 'Результат: '
mov eax,edi                     ; вызов подпрограммы печати значения
call iprintLF                   ; из 'edi' в виде символов

mov eax, rem                    ; вызов подпрограммы печати
call sprint                     ; сообщения 'Остаток от деления: '
mov eax,edx                     ; вызов подпрограммы печати значения
call iprintLF                   ; из 'edx' (остаток) в виде символов

call quit                       ; вызов подпрограммы завершения
```

Рис. 2.18: Отредактированный текст файла lab7-3.asm

```
dvshilonosov@dvshilonosov-RUDN:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
dvshilonosov@dvshilonosov-RUDN:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-3 lab7-3.o
dvshilonosov@dvshilonosov-RUDN:~/work/arch-pc/lab07$ ./lab7-3
Результат: 5
Остаток от деления: 1
```

Рис. 2.19: Создание и запуск исполняемого файла отредактированного текста lab7-3.asm

В качестве другого примера рассмотрим программу вычисления варианта задания по номеру студенческого билета, работающую по следующему алгоритму:

- вывести запрос на введение No 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.

В данном случае число, над которым необходимо проводить арифметические операции, вводится с клавиатуры. Как отмечалось выше ввод с клавиатуры осуществляется в символьном виде и для корректной работы арифметических операций в NASM символы необходимо преобразовать в числа. Для этого может быть использована функция atoi из файла in_out.asm.

Создадим файл variant.asm в каталоге ~/work/arch-pc/lab07. (рис. 2.20, 2.21)

Рис. 2.20: Создание файла variant.asm

```
/home/dvshilonosov/work/arch-pc/lab07/variant.asm
;------------------------------
; Программа вычисления варианта
;------------------------------

%include 'in_out.asm'

SECTION .data
msg: DB 'Введите No 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. 2.21: Текст программы из листинга 7.4. в файле variant.asm

Создадим исполняемый файл и запустим его. (рис. 2.22)

```
dvshilonosov@dvshilonosov-RUDN:~/work/arch-pc/lab07$ nasm -f elf variant.asm
dvshilonosov@dvshilonosov-RUDN:~/work/arch-pc/lab07$ ld -m elf_i386 -o variant variant.o
dvshilonosov@dvshilonosov-RUDN:~/work/arch-pc/lab07$ ./variant
Введите No студенческого билета:
1132221810
Ваш вариант: 11
```

Рис. 2.22: Создание и запуск исполняемого файла текста variant.asm

Проверим результат работы программы, вычислив номер варианта аналитически: (1132221810 % 20) + 1 = 10 + 1 = 11. Вариант по номеру студенческого билета

вычислен верно.

3 Ответы на вопросы

1. Q: Какие строки листинга 7.4 отвечают за вывод на экран сообщения “Ваш вариант:”?

A: Строки “rem: DB ‘Ваш вариант:’,0”, “mov eax, rem” и “call sprint” отвечают за вывод на экран сообщения ‘Ваш вариант:’. (рис. 3.1)

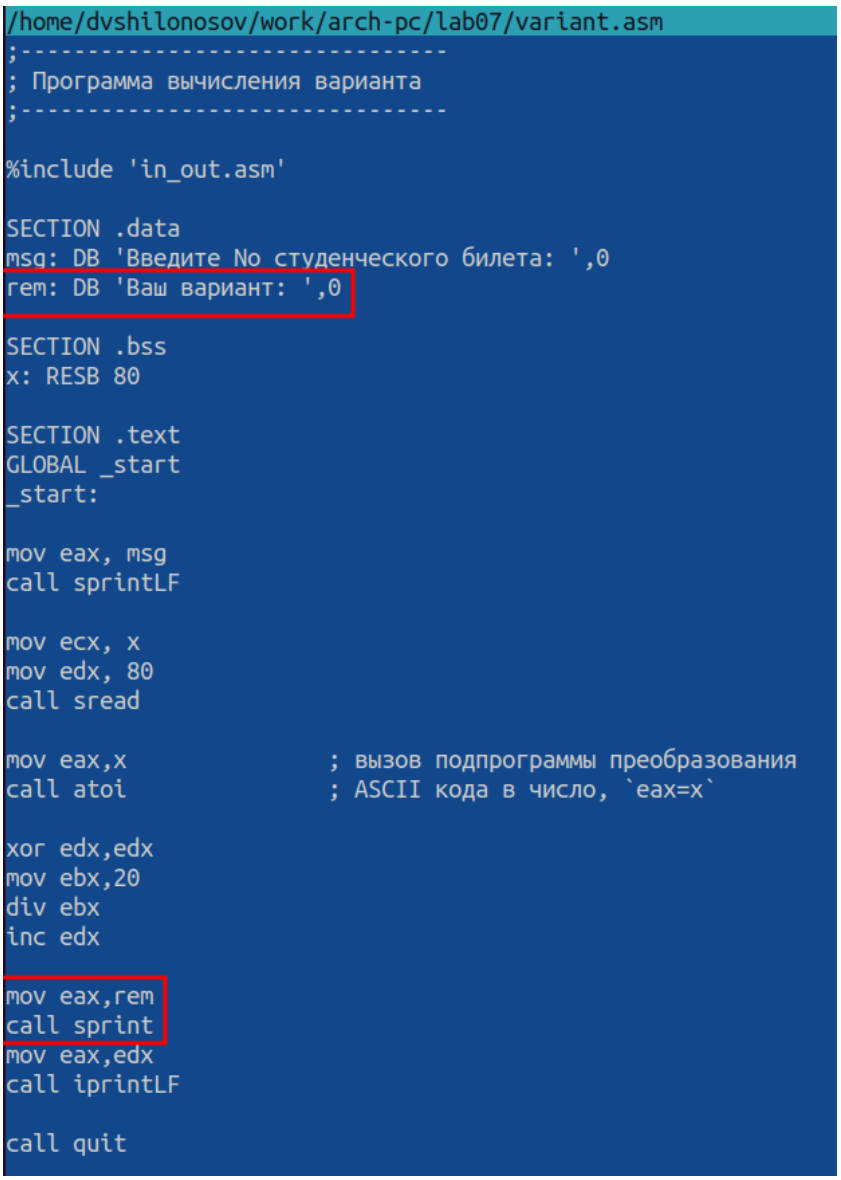

Рис. 3.1: Строки, отвечающие за вывод на экран сообщения “Ваш вариант:”

2. Q: Для чего используется следующие инструкции nasm?

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

A: 1) Инструкция mov ecx, x используется для передачи регистру ecx значение из переменной неинициализированной памяти x. 2) Инструкция mov edx,

80 используется для передачи регистру edx значение 80 непосредственно. 3) Инструкция call sread используется для ввода сообщения с клавиатуры.

3. Q: Для чего используется инструкция “call atoi”?

A: Функция atoi используется для преобразовании ASCII-кода символа в целое число.

4. Q: Какие строки листинга 7.4 отвечают за вычисления варианта?

A: Строки “xor edx, edx”, “mov ebx, 20”, “div ebx”, “inc edx” отвечают за вычисление варианта. (рис. 3.2)

```
/home/dvshilonosov/work/arch-pc/lab07/variant.asm
;-----------------------------
; Программа вычисления варианта
;-----------------------------

%include 'in_out.asm'

SECTION .data
msg: DB 'Введите No 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. 3.2: Строки, отвечающие за вычисление варианта”

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

A: Остаток от деления при выполнении инструкции “div ebx” записывается в регистр EDX.

6. Q: Для чего используется инструкция “inc edx”?

A: Инструкция “inc edx” используется для того, чтобы инкрементировать (увеличить на едиинцу) значение, хранящееся в регистре EDX.

7. Q: Какие строки листинга 7.4 отвечают за вывод на экран результата вычислений? (рис. 3.3)

A: Строки “mov eax,rem”, “call sprint”, “mov eax,edx”, “call iprintLF” отвечают за вывод на экран результата вычислений.

```
/home/dvshilonosov/work/arch-pc/lab07/variant.asm
;-----------------------------
; Программа вычисления варианта
;-----------------------------

%include 'in_out.asm'

SECTION .data
msg: DB 'Введите No 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. 3.3: Строки, отвечащие за вывод на экран результата вычислений

4 Выполнение задания для самостоятельной работы

Напишем 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. (рис. 4.1)

Номер варианта	Выражение для $f(x)$	x_1	x_2
11	$10(x+1)-10$	1	7

Рис. 4.1: Вариант 11

- При выполнении задания преобразовывать (упрощать) выражения для f(x) нельзя. При выполнении деления в качестве результата можно использовать только целую часть от деления и не учитывать остаток (т.е. 5 : 2 = 2).

Создадим файл task7-1.asm в каталоге ~/work/arch-pc/lab07. (рис. 4.2)

```
dvshilonosov@dvshilonosov-RUDN:~/work/arch-pc/lab07$ touch task7-1.asm
```

Рис. 4.2: Создание файла task7-1.asm

Отредактируем файл task7-1.asm. (рис. 4.3)

```
/home/dvshilonosov/work/arch-pc/lab07/task7-1.asm
%include 'in_out.asm'

SECTION .data

foo: DB 'f(x) = 10(x + 1) - 10', 0
msg: DB 'Введите значение x: ', 0
div: DB 'Результат: ', 0

SECTION .bss
x: RESB 80

SECTION .text
GLOBAL _start
_start:

mov eax, foo
call sprintLF

mov eax, msg
call sprint

mov ecx, x
mov edx, 80
call sread

mov eax, x
call atoi

add eax, 1

mov ebx, 10
imul ebx

sub eax, 10

mov edi, eax

mov eax, div
call sprint
mov eax, edi
call iprintLF

call quit
```

Рис. 4.3: Текст программы файла task7-1.asm

Создадим исполняемый файл и проверим его работу для значений x1 = 1 и x2

= 7. (рис. 4.4)

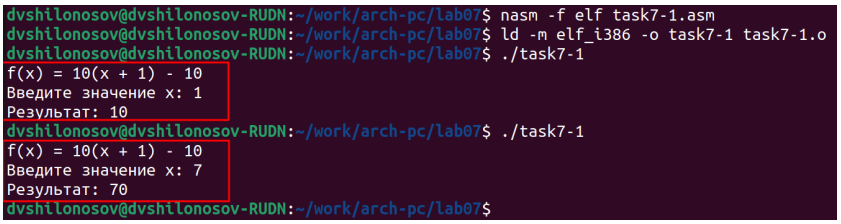

Рис. 4.4: Создание и запуск исполняемого файла текста task7-1.asm

5 Выводы

В процессе выполнения лабораторной работы были приобретены практичекие навыки работы с арифметическими инструкциями языка ассемблера NASM.