

Отчет по лабораторной работе №5

Архитектура компьютера

Никифоров Захар Сергеевич

Содержание

1	Цель работы	5
2	Порядок выполнения работы	6
2.1	Символьные и численные данные в NASM	6
2.2	Выполнение арифметических операций в NASM	10
3	Задание для самостоятельной работы	16

Список иллюстраций

2.1	Скриншот терминала тс 1	6
2.2	Скриншот терминала тс 2	6
2.3	Скриншот терминала тс 3	7
2.4	Скриншот терминала тс 4	7
2.5	Скриншот терминала тс 5	8
2.6	Скриншот терминала тс 6	8
2.7	Скриншот терминала тс 7	8
2.8	Скриншот терминала тс 8	9
2.9	Скриншот терминала тс 9	9
2.10	Скриншот терминала тс 10	9
2.11	Скриншот терминала тс 11	10
2.12	Скриншот терминала тс 12	10
2.13	Скриншот терминала тс 13	11
2.14	Скриншот терминала тс 14	11
2.15	Скриншот терминала тс 15	12
2.16	Скриншот терминала тс 16	12
2.17	Скриншот терминала тс 17	12
2.18	Скриншот терминала тс 18	13
2.19	Скриншот терминала тс 19	13
2.20	Скриншот терминала тс 20	14
2.21	Скриншот терминала тс 21	14
2.22	Скриншот терминала тс 22	15
3.1	Скриншот терминала тс 23	16
3.2	Скриншот терминала тс 24	16
3.3	Скриншот терминала тс 25	17
3.4	Скриншот терминала тс 26	17
3.5	Скриншот терминала тс 27	17

Список таблиц

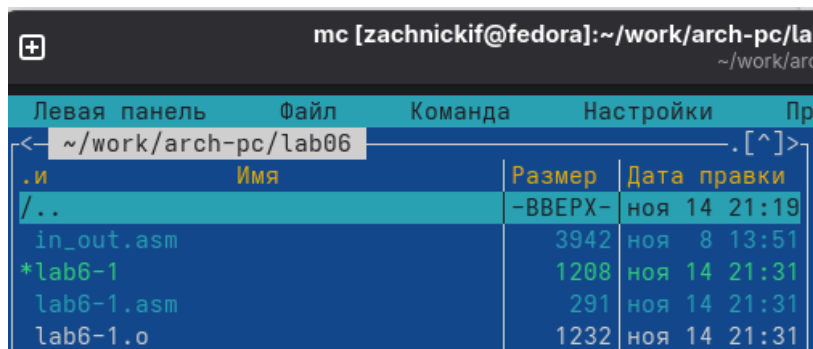
1 Цель работы

Освоение арифметических инструкций языка ассемблера NASM.

2 Порядок выполнения работы

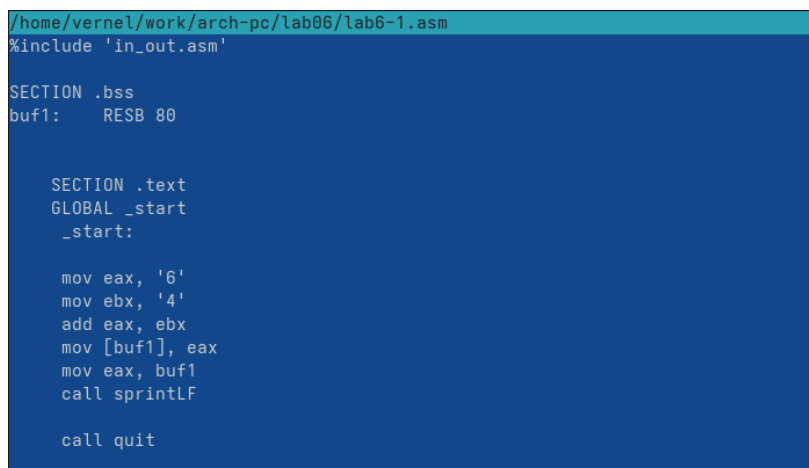
2.1 Символьные и численные данные в NASM

Создаем каталог *lab06*, а в нем создаем файл *lab6-1.asm*, записываем код из *Листинг 6.1* и компилируем его.



Левая панель	Файл	Команда	Настройки	Пр
< ~/work/arch-pc/lab06 .[^]>				
.и	Имя	Размер	Дата правки	
/..		-ВВЕРХ-	ноя 14 21:19	
	in_out.asm	3942	ноя 8 13:51	
	*lab6-1	1208	ноя 14 21:31	
	lab6-1.asm	291	ноя 14 21:31	
	lab6-1.o	1232	ноя 14 21:31	

Рисунок 2.1: Скриншот терминала mc 1



```
/home/vernel/work/arch-pc/lab06/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 sprintf
    call quit
```

Рисунок 2.2: Скриншот терминала mc 2

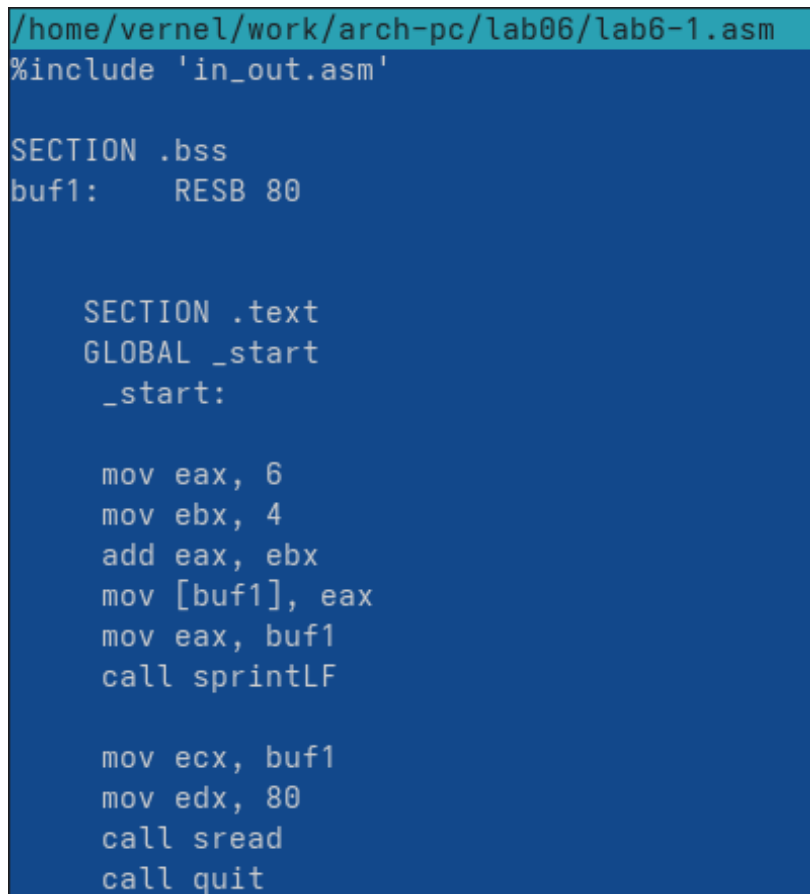


```
zachnickif@fedora:~/work/arch-pc/lab06$ ./lab6-1
j
```

Рисунок 2.3: Скриншот терминала тс 3

Получаем на вывод букву *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*.

Далее изменим код программы, убрав кавычки вокруг цифр.



```
/home/vernel/work/arch-pc/lab06/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

    mov ecx, buf1
    mov edx, 80
    call sread
    call quit
```

Рисунок 2.4: Скриншот терминала тс 4

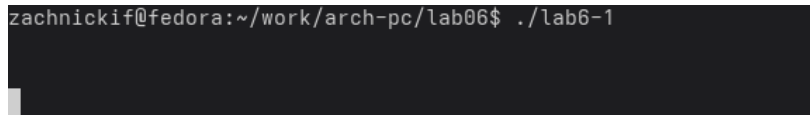


Рисунок 2.5: Скриншот терминала mc 5

На выход мы получаем символ переноса строки, который имеет код 10.

10	12	0x0A	1010	LF, \n
----	----	------	------	--------

Рисунок 2.6: Скриншот терминала mc 6

Символ не выражается в качестве *n* с обратной дробью, но видно перенос строки.

Теперь создадим новый файл *lab6-2.asm* запишем в него код из *Листинг 6.2* и скомпилируем его.

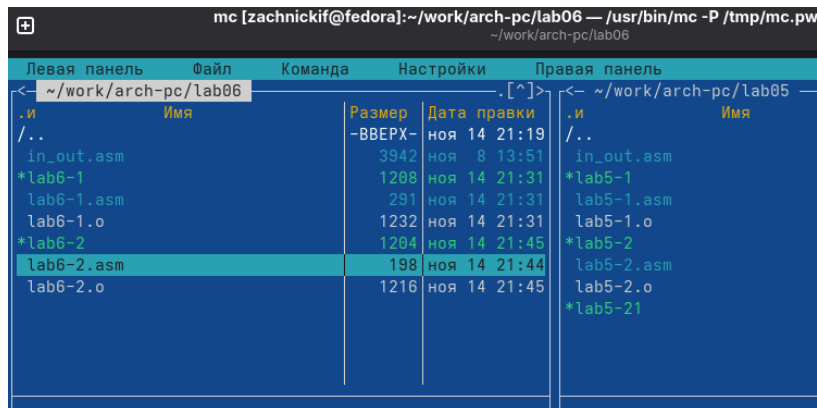
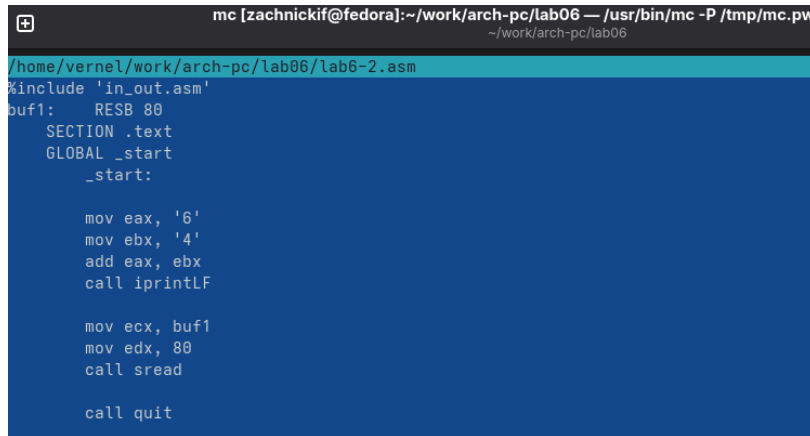


Рисунок 2.7: Скриншот терминала mc 7



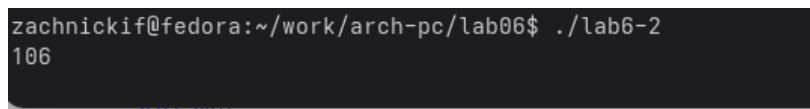
```
mc [zachnickif@fedora]:~/work/arch-pc/lab06 — /usr/bin/mc -P /tmp/mc.pw
~/work/arch-pc/lab06
/home/vernel/work/arch-pc/lab06/lab6-2.asm
%include 'in_out.asm'
buf1:  RESB 80
SECTION .text
GLOBAL _start
_start:

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

    mov ecx, buf1
    mov edx, 80
    call sread

    call quit
```

Рисунок 2.8: Скриншот терминала mc 8

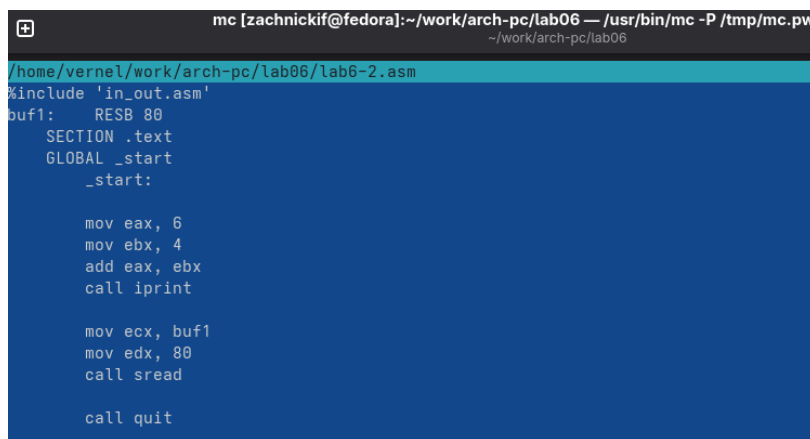


```
zachnickif@fedora:~/work/arch-pc/lab06$ ./lab6-2
106
```

Рисунок 2.9: Скриншот терминала mc 9

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 *Листинг 6.1*, функция `iprintLF` позволяет вывести число, а не символ, кодом которого является это число.

Теперь внесем изменения, убрав кавычки и заменив `iprintLF` на `iprint`.



```
mc [zachnickif@fedora]:~/work/arch-pc/lab06 — /usr/bin/mc -P /tmp/mc.pw
~/work/arch-pc/lab06
/home/vernel/work/arch-pc/lab06/lab6-2.asm
%include 'in_out.asm'
buf1:  RESB 80
SECTION .text
GLOBAL _start
_start:

    mov eax, 6
    mov ebx, 4
    add eax, ebx
    call iprint

    mov ecx, buf1
    mov edx, 80
    call sread

    call quit
```

Рисунок 2.10: Скриншот терминала mc 10

```
zachnickif@fedora:~/work/arch-pc/lab06$ ./lab6-2
10
```

Рисунок 2.11: Скриншот терминала тс 11

В результате работы программы мы получим число 10. Из отличий заметно отсутствие переноса строки.

2.2 Выполнение арифметических операций в NASM

Создадим файл *lab6-3.asm*, запишем в него код из *Листинг 6.3* и скомпилируем его.

Левая панель	Файл	Команда	Настройки	П
< ~/work/arch-pc/lab06 .[^]>				
.и	Имя	Размер	Дата правки	
./..		-ВВЕРХ-	ноя 14 21:19	
in_out.asm		3942	ноя 8 13:51	
*lab6-1		1208	ноя 14 21:31	
lab6-1.asm		291	ноя 14 21:31	
lab6-1.o		1232	ноя 14 21:31	
*lab6-2		1204	ноя 14 23:54	
lab6-2.asm		192	ноя 14 23:54	
lab6-2.o		1216	ноя 14 23:54	
*lab6-3		5204	ноя 14 22:12	
lab6-3.asm		525	ноя 14 22:12	
lab6-3.o		1552	ноя 14 22:12	

Рисунок 2.12: Скриншот терминала тс 12

```
mc [zachnickif@fedora]:~/work/arch-pc/lab06 — /usr/bin/mc -P /tmp/mc.pwd.3RXvDM
~/work/arch-pc/lab06
/home/vernel/work/arch-pc/lab06/lab6-3.asm 525/525
#include 'in_out.asm'

SECTION .data
div: DB 'Результат: ', 0
rem: DB 'Остаток от деления: ', 0
buf1:  RESB 80
SECTION .TEXT
GLOBAL _start
_start:

    mov eax, 5
    mov ebx, 2
    mul ebx
    add eax, 3
    xor edx, edx
    mov ebx, 3
    div ebx

    mov edi, eax

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

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

    mov ecx, buf1
    mov edx, 80
    call sread

    call quit
```

Рисунок 2.13: Скриншот терминала тс 13

```
zachnickif@fedora:~/work/arch-pc/lab06$ ./lab6-3
Результат: 4
Остаток от деления: 1
```

Рисунок 2.14: Скриншот терминала тс 14

Результат такой, какой и должен быть. Теперь внесем изменения, изменив функцию на $f(x) = (46+2)/5$.

```
mc [zachnickif@fedora]:~/work/arch-pc/lab06 — /usr/bin/mc -P /tmp/mc.pwd.3RXvDM
~/work/arch-pc/lab06
lab6-3.asm [-M--] 11 L: [ 8+ 9 17/ 36] *(283 / 525b) 0010 0x00A
GLOBAL _start
_start:
....
mov eax, 4
mov ebx, 6
mul ebx
add eax, 2
xor edx, edx
mov ebx, 5
div ebx
....
mov edi, eax
....
mov eax, div
call sprint
mov eax, edi
call iprintlf
```

Рисунок 2.15: Скриншот терминала тс 15

```
zachnickif@fedora:~/work/arch-pc/lab06$ ./lab6-3
Результат: 5
Остаток от деления: 1
```

Рисунок 2.16: Скриншот терминала тс 16

Результат изменился соответственно функции.

Теперь создадим файл *variant.asm*, запишем в него код из *Листинг 6.4* и скомпилируем.

*variant	5168	ноя 14 22:58
variant.asm	596	ноя 14 22:56
variant.o	1568	ноя 14 22:56

Рисунок 2.17: Скриншот терминала тс 17

```
mc [zachnickif@fedora]:~/work/arch-pc/lab06 — /usr/bin/mc -P /tmp/mc.pwd.3RXvDM
~/work/arch-pc/lab06
variant.asm [-M--] 8 L:[ 1+ 7 8/ 37] *(208 / 596b) 0082 0x052
%include 'in_out.asm'
SECTION .data
msg: DB 'Введите № студентского билета: ', 0
rem: DB 'Ваш вариант: ', 0
buf1: RESB 80
....
SECTION .bss
x:<> RESB 80
SECTION .text
GLOBAL _start
_start:
....
mov eax, msg
call sprintf
....
mov ecx, x
mov edx, 80
call sread
....
mov eax, x
call atoi
....
xor edx, edx
mov ebx, 20
div ebx
inc edx
....
mov eax, rem
call sprintf
mov eax, edx
call iprintLF
....
mov ecx, buf1
mov edx, 80
call sread
....
call quit
```

Рисунок 2.18: Скриншот терминала тс 18

```
zachnickif@fedora:~/work/arch-pc/lab06$ ./variant
Введите № студентского билета:
1032253520
Ваш вариант: 1
```

Рисунок 2.19: Скриншот терминала тс 19

В результате я получил 1. Это соответствует функции, так как мой номер студенческого билета кратен 20 и имеет остаток 0, а вариант 0 быть не может, поэтому программа превратила 0 в 1.

1. Следующие строки:

```
mov eax, msg  
call sprintf
```

Рисунок 2.20: Скриншот терминала mc 20

2. Чтобы прочитать номер студенческого билета, как число.
3. Эта функция нужна, чтобы преобразовать строку в число.
4. Следующие строки:

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

Рисунок 2.21: Скриншот терминала mc 21

5. В *EDX*, причем всегда.
6. Эта инструкция нужна, чтобы увеличить остаток на 1. В нашем случае помогает начать счёт варианта с 1, а не 0.
7. Следующие строки:

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

Рисунок 2.22: Скриншот терминала mc 22

3 Задание для самостоятельной работы

Создам файл *practice.asm* и, опираясь на прошлые, напишу следующий код, который будет вычислять функцию $f(x) = (10+2x)/3$ из введенного x , согласно моему варианту:

*practice	5064	ноя 14 23:41
practice.asm	641	ноя 14 23:24
practice.o	1472	ноя 14 23:24

Рисунок 3.1: Скриншот терминала mc 23

```
mc [zachnickif@fedora]:~/work/arch-pc/lab06 — /usr/bin/mc -P /tmp/mc.pwd.3RXvDM
~/work/arch-pc/lab06
/home/vernel/work/arch-pc/lab06/practice.asm 496/641
%include 'in_out.asm'

SECTION .data
exp: DB 'f(x) = (10+2x) / 3', 0
msg: DB 'Введите x: ', 0
res: DB 'Результат: ', 0

SECTION .bss
buf1: RESB 80

SECTION .text
GLOBAL _start
_start:
```

Рисунок 3.2: Скриншот терминала mc 24


```

_start:
    mov eax, exp
    call sprintf

    mov eax, msg
    call sprintf

    mov ecx, buf1
    mov edx, 80
    call sread

    mov eax, buf1
    call atoi

    mov ebx, 2
    mul ebx

    add eax, 10

    xor edx, edx
    mov ebx, 3
    div ebx

    mov edi, eax

    mov eax, res
    call sprintf

    mov eax, edi
    call iprintf

    mov ecx, buf1
    mov edx, 80
    call sread

    call quit

```

Рисунок 3.3: Скриншот терминала тс 25

Используя значения x равные 1 и 10.

```

zachnickif@fedora:~/work/arch-pc/lab06$ ./practice
f(x) = (10+2x) / 3
Введите x:
10
Результат: 10

```

Рисунок 3.4: Скриншот терминала тс 26

```

zachnickif@fedora:~/work/arch-pc/lab06$ ./practice
f(x) = (10+2x) / 3
Введите x:
1
Результат: 4

```

Рисунок 3.5: Скриншот терминала тс 27

Программа работает корректно, проверив аналитически, я пришел к выводу, что

и арифметически корректно тоже. # **Выводы**

В ходе работы были освоены арифметические конструкции языка *nasm*.