

Отчёт по лабораторной работе №6

Архитектура компьютера

Агапова Анна Антоновна

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Ответы на вопросы	10
4	Задание для самостоятельной работы	11
5	Выводы	13
6	Список литературы	14

Список иллюстраций

2.1	Создание каталога и файла	6
2.2	Ввод текста программы	6
2.3	Запуск исполняемого файла.....	6
2.4	Исправление текста программы	7
2.5	Запуск исполняемого файла.....	7
2.6	Создание файла	7
2.7	Ввод текста программы	7
2.8	Запуск исполняемого файла.....	7
2.9	Изменение текста программы	8
2.10	Запуск исполняемого файла.....	8
2.11	Изменение текста программы	8
2.12	Запуск исполняемого файла.....	8
2.13	Создание файла.....	8
2.14	Ввод текста программы	9
2.15	Запуск исполняемого файла.....	9
2.16	Изменение текста программы	9
2.17	Запуск исполняемого файла.....	9
2.18	Создание файла.....	10
2.19	Ввод текста программы	10
2.20	Запуск исполняемого файла.....	10
2.21	Создание файла.....	11
2.22	Ввод текста программы	11
2.23	Запуск исполняемого файла.....	12
2.24	Запуск исполняемого файла.....	12

Список таблиц

1 Цель работы

Освоение арифметических инструкций языка ассемблера NASM.

2 Выполнение лабораторной работы

1. Создаю каталог для программ лабораторной работы № 6, перехожу в него и создаю файл lab6-1.asm. (рис. 2.1).

```
aaagapova@astra:~$ mkdir ~/work/arch-pc/lab06
aaagapova@astra:~$ cd ~/work/arch-pc/lab06
aaagapova@astra:~/work/arch-pc/lab06$ touch lab6-1.asm
```

Рисунок 2.1: Создание каталога и файла

2. Ввожу в файл lab6-1.asm текст программы из листинга 6.1. (рис. 2.2).

```
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
|
```

Рисунок 2.2: Ввод текста программы

3. Создаю исполняемый файл и запускаю его. (рис. 2.3)

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

Рисунок 2.3: Запуск исполняемого файла

4. Далее изменю текст программы и вместо символов, запишу в регистры числа. (рис. 2.4)

```
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 2.4: Исправление текста программы

5. Создаю исполняемый файл и запускаю его. (рис. 2.5).

```
aaagapova@astra:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
aaagapova@astra:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
aaagapova@astra:~/work/arch-pc/lab06$ ./lab6-1
```

Рисунок 2.5: Запуск исполняемого файла

6. Создаю файл lab6-2.asm в каталоге ~/work/arch-pc/lab06 и проверяю его наличие. (рис. 2.6)

```
aaagapova@astra:~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/lab6-2.asm
aaagapova@astra:~/work/arch-pc/lab06$ ls
in_out.asm lab6-1 lab6-1.asm lab6-1.o lab6-2.asm
```

Рисунок 2.6: Создание файла

7. Ввожу в созданный файл текст программы из листинга 6.2. (рис. 2.7)

```
lab6-2.asm
#include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,'6'
mov ebx,'4'
add eax,ebx
call iprintLF
call quit
```

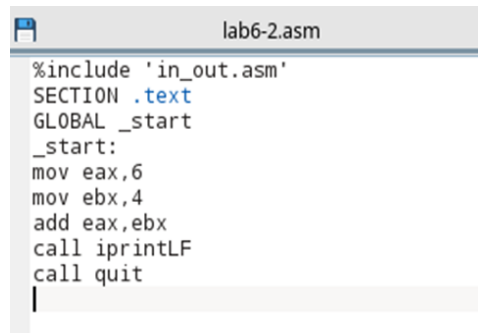
Рисунок 2.7: Ввод текста программы

8. Создаю исполняемый файл и запускаю его. (рис. 2.8)

```
aaagapova@astra:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
aaagapova@astra:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
aaagapova@astra:~/work/arch-pc/lab06$ ./lab6-2
106
```

Рисунок 2.8: Запуск исполняемого файла

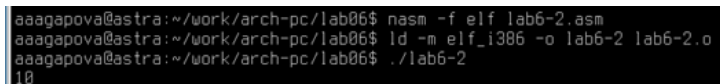
9. Аналогично предыдущему примеру изменю символы на числа. (рис. 2.9)



```
lab6-2.asm
#include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,6
mov ebx,4
add eax,ebx
call iprintLF
call quit
|
```

Рисунок 2.9: Изменение текста программы

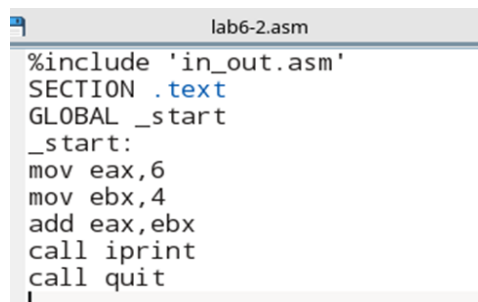
10. Создаю исполняемый файл и запускаю его. (рис. 2.10).



```
aaagapova@astra:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
aaagapova@astra:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
aaagapova@astra:~/work/arch-pc/lab06$ ./lab6-2
18
```

Рисунок 2.10: Запуск исполняемого файла

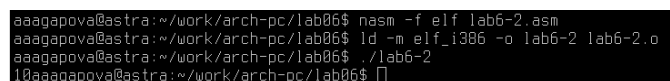
11. Заменяю функцию iprintLF на iprint. (рис. 2.11).



```
lab6-2.asm
#include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,6
mov ebx,4
add eax,ebx
call iprint
call quit
|
```

Рисунок 2.11: Изменение текста программы

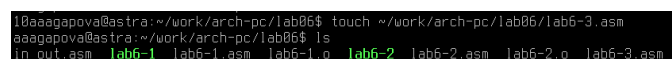
12. Создаю исполняемый файл и запускаю его. (рис. 2.12).



```
aaagapova@astra:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
aaagapova@astra:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
aaagapova@astra:~/work/arch-pc/lab06$ ./lab6-2
10aaagapova@astra:~/work/arch-pc/lab06$
```

Рисунок 2.12: Запуск исполняемого файла

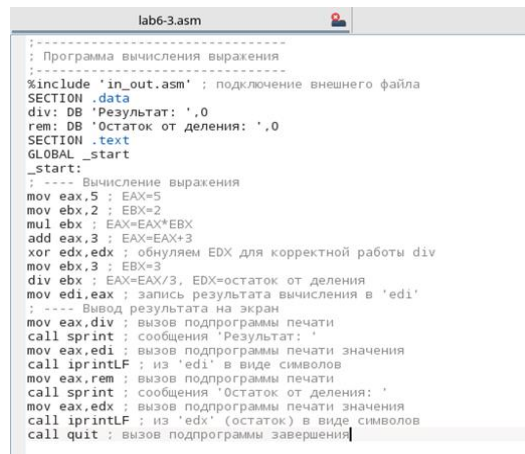
13. Создаю файл lab6-3.asm в каталоге ~/work/arch-pc/lab06 и проверяю его наличие. (рис. 2.13).



```
10aaagapova@astra:~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/lab6-3.asm
aaagapova@astra:~/work/arch-pc/lab06$ ls
in_out.asm  lab6-1  lab6-1.asm  lab6-1.o  lab6-2  lab6-2.asm  lab6-2.o  lab6-3.asm
```

Рисунок 2.13: Создание файла

14. Внимательно изучаю текст программы из листинга 6.3 и ввожу в lab6-3.asm для вычисления арифметического выражения $\square(\square) = (5 * 2 + 3)/3$. (рис. 2.14).



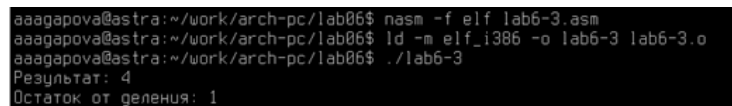
```

lab6-3.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

```

Рисунок 2.14: Ввод текста программы

15. Создаю исполняемый файл и запускаю его. (рис. 2.15).



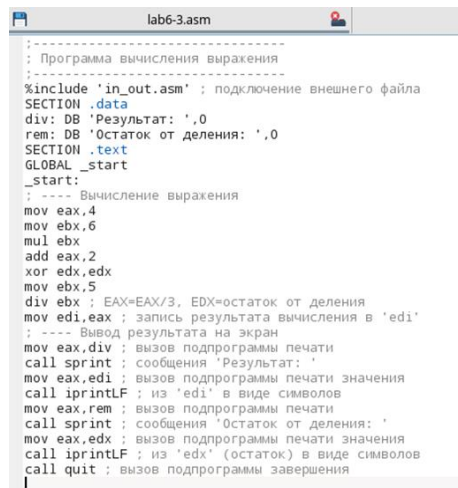
```

aaagapova@astra:~/work/arch-pc/lab06$ nasm -f elf lab6-3.asm
aaagapova@astra:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-3 lab6-3.o
aaagapova@astra:~/work/arch-pc/lab06$ ./lab6-3
Результат: 4
Остаток от деления: 1

```

Рисунок 2.15: Запуск исполняемого файла

16. Меняю текст программы для вычисления выражения $\square(\square) = (4 * 6 + 2)/5$. (рис. 2.16).



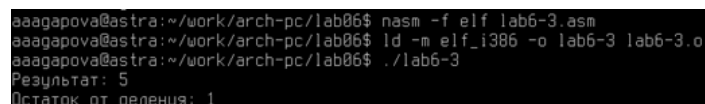
```

lab6-3.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
mov ebx,6
mul ebx
add eax,2
xor edx,edx
mov ebx,5
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

```

Рисунок 2.16: Изменение текста программы

17. Создаю исполняемый файл и запускаю его. (рис. 2.17)



```

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

```

Рисунок 2.17: Запуск исполняемого файла

18. Создаю файл variant.asm в каталоге ~/work/arch-pc/lab06 и проверяю его наличие. (рис. 2.18)

```
aaagapova@astra:~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/variant.asm
aaagapova@astra:~/work/arch-pc/lab06$ ls
in_out.asm lab6-1 lab6-1.o lab6-2 lab6-2.o lab6-3 lab6-3.o variant.asm
```

Рисунок 2.18: Создание файла

19. Внимательно изучаю текст программы из листинга 6.4 и ввожу в файл variant.asm. (рис. 2.19).

```
variant.asm
;-----
; 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 sprintf
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
```

Рисунок 2.19: Ввод текста программы

20. Создаю исполняемый файл и запускаю его. (рис. 2.20).

```
aaagapova@astra:~/work/arch-pc/lab06$ nasm -f elf variant.asm
aaagapova@astra:~/work/arch-pc/lab06$ ld -m elf_i386 -o variant variant.o
aaagapova@astra:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1032251933
Ваш вариант: 14
```

Рисунок 2.20: Запуск исполняемого файла

3 Ответы на вопросы

1. За вывод сообщения “Ваш вариант” отвечают строки кода:

```
mov eax, rem
call sprint
```

2. Инструкция mov ecx, x используется, чтобы поместить адрес вводимой строки x в регистр ecx, mov edx, 80 - задаёт максимальную длину ввода, call sread - вызывает функцию чтения строки с клавиатуры.

3. За call atoi используется для преобразования строки ASCII-коды в число и записывает результат в eax.

4. За вычисления варианта отвечают строки:

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

5. Остаток записывается в регистр edx.

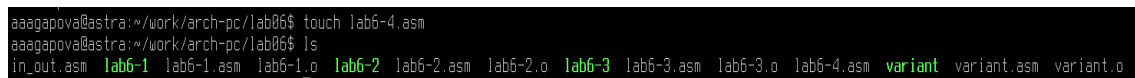
6. inc edx используется для увеличения остатка на 1.

7. За вывод на экран результатов вычислений отвечают строки:

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

4 Задание для самостоятельной работы

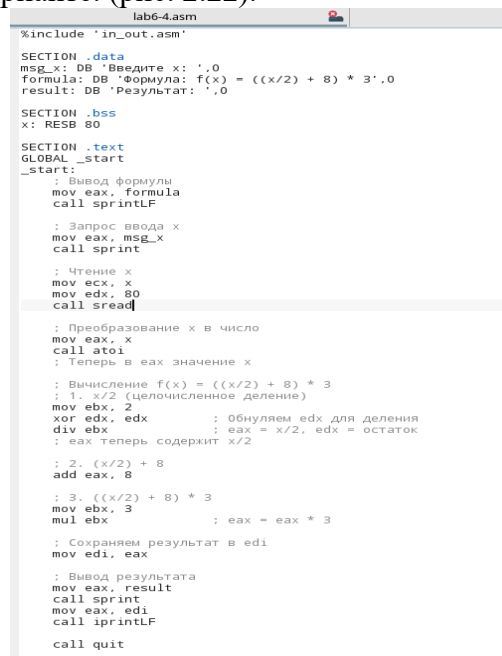
1. Создаю файл lab6-4.asm в каталоге ~/work/arch-pc/lab06 и проверяю его наличие. (рис. 2.21).



```
aaagapova@astra:~/work/arch-pc/lab06$ touch lab6-4.asm  
aaagapova@astra:~/work/arch-pc/lab06$ ls  
in_out.asm lab6-1 lab6-1.asm lab6-1.o lab6-2 lab6-2.asm lab6-2.o lab6-3 lab6-3.asm lab6-3.o lab6-4.asm variant variant.asm variant.o
```

Рисунок 2.21: Создание файла

2. Ввожу в созданный файл программу для вычисления арифметического выражения, которое указано в 14 варианте. (рис. 2.22).



```
lab6-4.asm  
%include "in_out.asm"  
  
SECTION .data  
msg_x: DB "Введите x: ",0  
formula: DB "Формула: f(x) = ((x/2) + 8) * 3",0  
result: DB "Результат: ",0  
  
SECTION .bss  
x: RESB 80  
  
SECTION .text  
GLOBAL _start  
_start:  
; Вывод формулы  
mov eax, formula  
call sprintLf  
  
; Запрос ввода x  
mov eax, msg_x  
call sprint  
  
; Чтение x  
mov ecx, x  
mov edx, 80  
call sread  
  
; Преобразование x в число  
mov eax, x  
call atoi  
; Теперь в eax значение x  
  
; Вычисление f(x) = ((x/2) + 8) * 3  
; 1. x/2 (целочисленное деление)  
mov ebx, 2  
xor edx, edx ; Обнуляем edx для деления  
div ebx ; eax = x/2, edx = остаток  
; eax теперь содержит x/2  
  
; 2. (x/2) + 8  
add eax, 8  
  
; 3. ((x/2) + 8) * 3  
mov ebx, 3  
mul ebx ; eax = eax * 3  
  
; Сохраняем результат в edi  
mov edi, eax  
  
; Вывод результата  
mov eax, result  
call sprint  
mov eax, edi  
call iprintLf  
call quit
```

Рисунок 2.22: Ввод текста программы

3. Создаю исполняемый файл и запускаю его для $x1=1$. (рис. 2.23).

```
aaagapova@astra:~/work/arch-pc/lab06$ nasm -f elf lab6-4.asm
aaagapova@astra:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-4 lab6-4.o
aaagapova@astra:~/work/arch-pc/lab06$ ./lab6-4
Формула:  $f(x) = ((x/2) + 8) * 3$ 
Введите x: 1
Результат: 24
```

Рисунок 2.23: Запуск исполняемого файла

4. Создаю исполняемый файл и запускаю его для $x2=4$. (рис. 2.24).

```
aaagapova@astra:~/work/arch-pc/lab06$ nasm -f elf lab6-4.asm
aaagapova@astra:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-4 lab6-4.o
aaagapova@astra:~/work/arch-pc/lab06$ ./lab6-4
Формула:  $f(x) = ((x/2) + 8) * 3$ 
Введите x: 4
Результат: 30
```

Рисунок 2.24: Запуск исполняемого файла

5 Выводы

При выполнении данной лабораторной работы я освоила арифметические инструкции языка ассемблера NASM.

6 Список литературы

1. GDB: The GNU Project Debugger. — URL: <https://www.gnu.org/software/gdb/>.
2. GNU Bash Manual. — 2016. — URL: <https://www.gnu.org/software/bash/manual/>.
3. Midnight Commander Development Center. — 2021. — URL: <https://midnight-commander.org/>.
4. NASM Assembly Language Tutorials. — 2021. — URL: <https://asmtutor.com/>.
5. Newham C. Learning the bash Shell: Unix Shell Programming. — O'Reilly Media, 2005. — 354 с. — (In a Nutshell). — ISBN 0596009658. — URL: <http://www.amazon.com/Learningbash-Shell-Programming-Nutshell/dp/0596009658>.
6. Robbins A. Bash Pocket Reference. — O'Reilly Media, 2016. — 156 с. — ISBN 978-1491941591.
7. The NASM documentation. — 2021. — URL: <https://www.nasm.us/docs.php>.
8. Zarrelli G. Mastering Bash. — Packt Publishing, 2017. — 502 с. — ISBN 9781784396879.
9. Колдаев В. Д., Лупин С. А. Архитектура ЭВМ. — М. : Форум, 2018.
10. Куляс О. Л., Никитин К. А. Курс программирования на ASSEMBLER. — М. : Солон-Пресс,
11. Новожилов О. П. Архитектура ЭВМ и систем. — М. : Юрайт, 2016.
12. Расширенный ассемблер: NASM. — 2021. — URL: <https://www.opennet.ru/docs/RUS/nasm/>.
13. Робачевский А., Немнюгин С., Стесик О. Операционная система UNIX. — 2-е изд. — БХВПетербург, 2010. — 656 с. — ISBN 978-5-94157-538-1.
14. Столяров А. Программирование на языке ассемблера NASM для ОС Unix. — 2-е изд. — М. : МАКС Пресс, 2011. — URL: http://www.stolyarov.info/books/asm_unix.
15. Таненбаум Э. Архитектура компьютера. — 6-е изд. — СПб. : Питер, 2013. — 874 с. — (Классика Computer Science).
16. Таненбаум Э., Бос Х. Современные операционные системы. — 4-е изд. — СПб. : Питер, 2015. — 1120 с. — (Классика Computer Science).