

Отчет по лабораторной работе №6

дисциплина: Архитектура компьютера

Михайлова Регина Алексеевна

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выполнение заданий для самостоятельной работы	16
4	Вывод	19
5	Список литературы	20

Список иллюстраций

2.1	Создание каталога и файла	6
2.2	Запуск исполняемого файла	7
2.3	Измененный файл	8
2.4	Таблица ASCII	8
2.5	Создание и запуск файла	9
2.6	Запуск измененного файла	10
2.7	Создание и запуск файла	10
2.8	Запуск файла lab6-3	12
2.9	Измененный файл lab6-3	12
2.10	Запуск измененного файла lab6-3	13
2.11	Результат программы	14
3.1	Результат программы	17
3.2	Результат программы	18

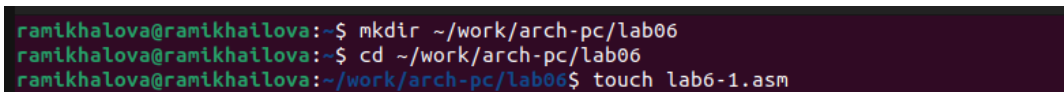
Список таблиц

1 Цель работы

Освоение арифметических инструкций языка ассемблера NASM.

2 Выполнение лабораторной работы

1. Создадим каталог для программ лабораторной работы № 6, перейдем в него и создадим файл lab6-1.asm (рис. 2.1).



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

Рис. 2.1: Создание каталога и файла

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

Введём в файл lab6-1.asm текст программы из листинга 6.1. В данной программе в регистр еах записываем символ 6 (mov еах, '6'), в регистр ебх символ 4 (mov ебх, '4'). Далее к значению в регистре еах прибавляем значение регистра ебх (add еах, ебх, результат сложения запишется в регистр еах). Далее выводим результат. Так как для работы функции sprintf в регистр еах должен быть записан адрес, необходимо использовать дополнительную переменную. Для этого запишем значение регистра еах в переменную buf1 (mov [buf1], еах), а затем запишем адрес переменной buf1 в регистр еах (mov еах, buf1) и вызовем функцию sprintf.

Листинг 6.1. Программа вывода значения регистра еах

```
%include 'in_out.asm'
SECTION .bss
```

```

buf1: RESB 80

SECTION .text

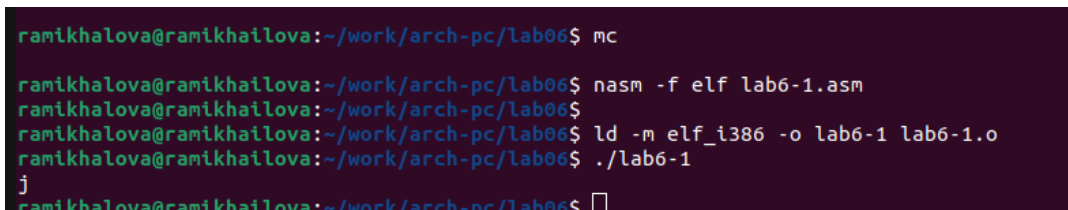
GLOBAL _start

_start:

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

```

Создаем исполняемый файл и запускаем его (рис. 2.2).



```

ramikhalova@ramikhallova:~/work/arch-pc/lab06$ mc
ramikhalova@ramikhallova:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
ramikhalova@ramikhallova:~/work/arch-pc/lab06$
ramikhalova@ramikhallova:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
ramikhalova@ramikhallova:~/work/arch-pc/lab06$ ./lab6-1
j
ramikhalova@ramikhallova:~/work/arch-pc/lab06$

```

Рис. 2.2: Запуск исполняемого файла

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, а код символа 4 – 00110100. Команда `add eax, ebx` запишет в регистр `eax` сумму кодов – 01101010 (106), что в свою очередь является кодом символа `j`.

3. Далее изменим текст программы и вместо символов, запишем в регистры числа. Ис- правим текст программы следующим образом (рис. 2.3):

заменяем строки: `mov eax, '6'` `mov ebx, '4'`
на строки: `mov eax, 6` `mov ebx, 4`

```

GNU nano 6.2 /home
#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

```

Рис. 2.3: Измененный файл

Создаем исполняемый файл и запускаем его.

Как и в предыдущем случае при исполнении программы мы не получим число 10. В данном случае выводится символ с кодом 10. Пользуясь таблицей ASCII определяем какому символу соответствует код 10 (рис. 2.4).

Таблица 12.1. Таблица символов ASCII

DEC	OCT	HEX	BIN	Symbol
0	0	0x00	0	NUL, \0
1	1	0x01	1	SOH
2	2	0x02	10	STX
3	3	0x03	11	ETX
4	4	0x04	100	EOT
5	5	0x05	101	ENQ
6	6	0x06	110	ACK
7	7	0x07	111	BEL
8	10	0x08	1000	BS
9	11	0x09	1001	HT, \t
10	12	0x0A	1010	LF, \n
11	13	0x0B	1011	VT
12	14	0x0C	1100	FF

Рис. 2.4: Таблица ASCII

Коду 10 соответствует символ переноса строки, поэтому мы видим на экране пустую строку.

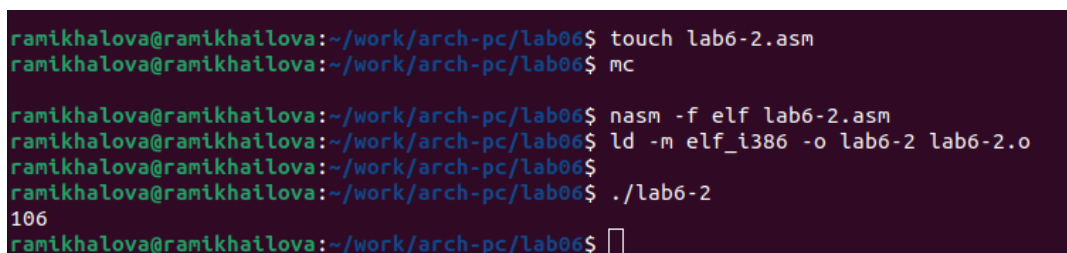
4. Для работы с числами в файле `in_out.asm` реализованы подпрограммы для преобразования ASCII символов в числа и обратно. Преобразуем текст программы из Листинга 6.1 с использованием этих функций.

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

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
```

Создаем исполняемый файл и запускаем его (рис. 2.5).



```
ramikhalova@ramikhallova:~/work/arch-pc/lab06$ touch lab6-2.asm  
ramikhalova@ramikhallova:~/work/arch-pc/lab06$ mc  
  
ramikhalova@ramikhallova:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm  
ramikhalova@ramikhallova:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o  
ramikhalova@ramikhallova:~/work/arch-pc/lab06$  
ramikhalova@ramikhallova:~/work/arch-pc/lab06$ ./lab6-2  
106  
ramikhalova@ramikhallova:~/work/arch-pc/lab06$
```

Рис. 2.5: Создание и запуск файла

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

5. Аналогично предыдущему примеру изменим символы на числа.

Заменяем строки: `mov eax,'6' mov ebx,'4'`

на строки: `mov eax,6 mov ebx,4`

Создаем исполняемый файл и запускаем его (рис. 2.6).

```

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

```

Рис. 2.6: Запуск измененного файла

А теперь заменим функцию `iprintLF` на `iprint`. Снова создаем исполняемый файл и запускаем его (рис. 2.7).

```

10
ramikhalova@ramikhailova:~/work/arch-pc/lab06$ mc

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

```

Рис. 2.7: Создание и запуск файла

Можем заметить, что функция `iprint` в отличие от `iprintLF` не выводит на экран перенос строки после числа 10.

- Ознакомимся с арифметическими операциями NASM. В качестве примера приведем программу вычисления арифметического выражения $f(x) = (5 * 2 + 3) : 3$.

Создаем файл `lab6-3.asm` в каталоге `~/work/arch-pc/lab06`.

Внимательно изучаем текст программы из листинга 6.3 и вводим в `lab6-3.asm`.

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

```

;-----
; 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

```

Создаем исполняемый файл и запускаем его (рис. 2.8).

```

lab6-3.asm:6: error: parser: instruction expected
ramikhalova@ramikhalova:~/work/arch-pc/lab06$ mc
ramikhalova@ramikhalova:~/work/arch-pc/lab06$ nasm -f elf lab6-3.asm
ramikhalova@ramikhalova:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-3 lab6-3.o
ramikhalova@ramikhalova:~/work/arch-pc/lab06$ ./lab6-3
Результат: 4
Остаток от деления: 1
ramikhalova@ramikhalova:~/work/arch-pc/lab06$ █

```

Рис. 2.8: Запуск файла lab6-3

Изменим текст программы для вычисления выражения $f(x) = (4 * 6 + 2) / 5$ (рис. 2.9). Создадим исполняемый файл и проверяем его работу (рис. ??).

```

GNU nano 0.2
%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 ; EAX=EAX*EBX
add eax,2 ;
xor edx,edx ; обнуляем EDX для корректной работы div
mov ebx,5 ;
div ebx ; EBX=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

```

Рис. 2.9: Измененный файл lab6-3

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

Рис. 2.10: Запуск измененного файла lab6-3

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: $(\text{№} \bmod 20) + 1$, где № – номер студенческого билета (В данном случае $\text{№} \bmod 20$ – это остаток от деления № на 20).
 - вывести на экран номер варианта.

Создаем файл variant.asm в каталоге ~/work/arch-pc/lab06.

Внимательно изучаем текст программы из листинга 6.4 и вводим в файл variant.asm.

Листинг 6.4. Программа вычисления варианта задания по номеру студенческого билета

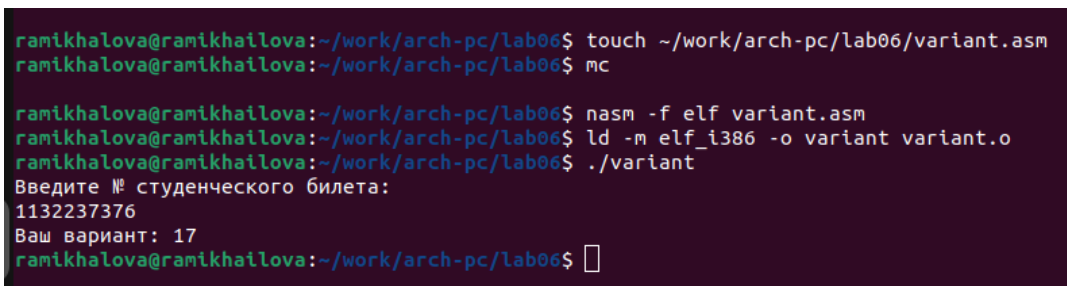
```
;-----
; Программа вычисления варианта
;-----
%include 'in_out.asm'
SECTION .data
msg: DB 'Введите № студенческого билета: ',0
rem: DB 'Ваш вариант: ',0
SECTION .bss
x: RESB 80
SECTION .text
GLOBAL _start
```

```

_start:
mov eax, msg
call 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 sprintf
mov eax,edx
call iprintLF
call quit

```

Создаем исполняемый файл и запускаем его (рис. 2.11).



```

ramikhalova@ramikhalilova:~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/variant.asm
ramikhalova@ramikhalilova:~/work/arch-pc/lab06$ mc
ramikhalova@ramikhalilova:~/work/arch-pc/lab06$ nasm -f elf variant.asm
ramikhalova@ramikhalilova:~/work/arch-pc/lab06$ ld -m elf_i386 -o variant variant.o
ramikhalova@ramikhalilova:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1132237376
Ваш вариант: 17
ramikhalova@ramikhalilova:~/work/arch-pc/lab06$ 

```

Рис. 2.11: Результат программы

Ответим на вопросы по лабораторной работе:

1. В листинге 6.4 за вывод на экран сообщения 'Ваш вариант:' отвечают строки:
 - rem: DB 'Ваш вариант:',0 ;в строке мы объявляем переменную rem, куда

- записали искомую строку • `mov eax,rem` ; помещаем строку в регистр `eax` • `call sprint` ; вызываем подпрограмму вывода из файла `in_out.asm`
2. Инструкции `mov ecx,x` -> `mov edx,80` -> `call sread` используются для того, чтобы ввести с клавиатуры строку отведённого размера (80) и поместить её по адресу `x`. Для этого `x` помещаем в регистр `ecx`, а длину строки (80) • в регистр `edx`. `call sread` - вызов функции печати.
3. Инструкция `call atoi` используется для преобразования символов в числа.
4. За вычисление варианта отвечают строки: • `mov eax,x` ; поместили `x` в регистр `eax` • `call atoi` ; преобразование символов в число • `xor edx,edx` ; обнуляем `edx` • `mov ebx,20` : поместили в регистр `ebx` число 20 • `div ebx` ; поделили число, лежащее в `eax`, на число, лежащее в `ebx` • `inc edx` ; `edx + 1`
5. Остаток от деления при выполнении `div ebx` записывается в регистр `edx`.
6. Инструкция `inc edx` используется для увеличения значения регистра `edx` на 1.
- 7.
8. За вывод на экран результата вычислений отвечают строки: • `mov eax,edx` ; помещаем результат вычислений в регистр `eax` • `call iprintLF` ; выводим на экран содержимое регистра `eax`

3 Выполнение заданий для самостоятельной работы

Необходимо написать программу, вычисляющую значение заданной функции $f(x)$ в зависимости от введённого значения x . Варианту 17 соответствует формула следующей функции: $/18(x + 1)/6$

Создаем программу (рис. 3.1).


```

GNU nano 0.2
%include 'in_out.asm'
SECTION .data
msg: DB 'Введите x:',0
rem: DB 'f(x) = ',0
SECTION .bss
x: RESB 80
SECTION .text
GLOBAL _start
_start:
;18(x+1)/6
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
inc eax
mov ebx,18
mul ebx
mov ebx,6
div ebx
mov edx,eax

mov eax,rem
call sprintf
mov eax,edx
call iprintfLF
call quit

```

Рис. 3.1: Результат программы

И проверяем его работу (рис. 3.2).

```
ramikhalova@ramikhalova:~/work/arch-pc/lab06$ nasm -f elf funk.asm
ramikhalova@ramikhalova:~/work/arch-pc/lab06$ ld -m elf_i386 -o funk funk.o
ramikhalova@ramikhalova:~/work/arch-pc/lab06$ ./funk
Введите x:
2
f(x) = 9
ramikhalova@ramikhalova:~/work/arch-pc/lab06$
```

Рис. 3.2: Результат программы

4 Вывод

Мы освоили арифметические инструкции языка ассемблера NASM, научились составлять арифметические программы.

5 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/Learning-bash-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 978-1787121111.
- 9.
10. Колдаев В. Д., Лупин С. А. Архитектура ЭВМ. — М. : Форум, 2018.
11. Куляс О. Л., Никитин К. А. Курс программирования на ASSEMBLER. — М. : Солон-Пресс