

Лабораторная работа номер 6.

Арифметические операции в NASM

Сорокин Кирилл

Содержание

1	Цель работы	4
2	Задание	5
3	Теоретическое введение	6
4	Выполнение лабораторной работы	7
4.1	Ответы на вопросы:	13
4.2	Самостоятельная работа	13
5	Выводы	16
	Список литературы	17

Список иллюстраций

4.1	Создание файлов и директорий	7
4.2	Текст первой программы	7
4.3	Первая попытка выполнить	8
4.4	Редактирование чисел	8
4.5	Вторая попытка выполнить	8
4.6	Редактирование второго файла	9
4.7	Попытка запуска второго файла	9
4.8	Изменение файла 2	9
4.9	Корректная работа программы	10
4.10	Содержимое файла lab6-3.asm	10
4.11	Выполнение программы вычисляющей $(5 * 2 + 3)/3$	11
4.12	Подстановка других значений	11
4.13	Выполнение изменённой программы	11
4.14	Мой вариант - 1	12
4.15	Создание файла sam.asm	13
4.16	Текст самостоятельной работы	14
4.17	Работа самостоятельной программы	15

1 Цель работы

Научиться писать арифметические инструкции языка ассемблер

2 Задание

Изучить приведённый материал на практике и выполнить самостоятельную работу.

3 Теоретическое введение

Схема команды целочисленного сложения `add` (от англ. `addition` - добавление) выполняет сложение двух операндов и записывает результат по адресу первого операнда. Команда целочисленного вычитания `sub` (от англ. `subtraction` – вычитание) работает аналогично команде `add`. Довольно часто при написании программ встречается операция прибавления или вычитания единицы. Прибавление единицы называется инкрементом, а вычитание декрементом. Для этих операций существуют специальные команды: `inc` (от англ. `increment`) и `dec` (от англ. `decrement`), которые увеличивают и уменьшают на 1 свой операнд.

4 Выполнение лабораторной работы

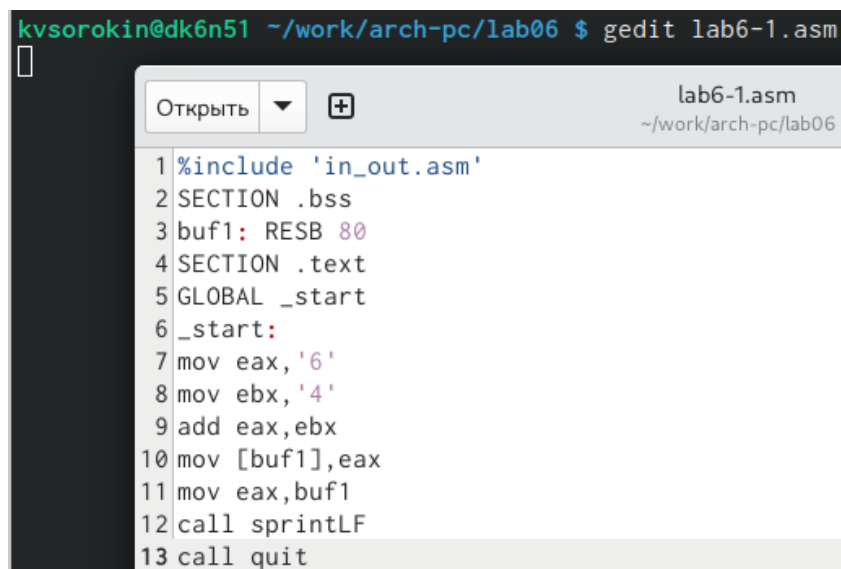
Создадим необходимые для работы директории и файлы (рис. 4.1).

```
kvsorokin@dk6n51 ~ $ mkdir ~/work/arch-pc/lab06
kvsorokin@dk6n51 ~ $ cd ~/work/arch-pc/lab06
kvsorokin@dk6n51 ~/work/arch-pc/lab06 $ touch lab6-1.asm
```

Рис. 4.1: Создание файлов и директорий

Откроем файл lab6-1.asm и введём в него текст программы(рис. 4.2).

```
kvsorokin@dk6n51 ~/work/arch-pc/lab06 $ gedit lab6-1.asm
```



```
1 %include 'in_out.asm'
2 SECTION .bss
3 buf1: RESB 80
4 SECTION .text
5 GLOBAL _start
6 _start:
7 mov eax, '6'
8 mov ebx, '4'
9 add eax, ebx
10 mov [buf1], eax
11 mov eax, buf1
12 call sprintLF
13 call quit
```

Рис. 4.2: Текст первой программы

После компиляции файлов запустим программу и увидим, нежелаемый результат - j, вместо 10(рис. 4.3).

```

kvsorokin@dk6n51 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
kvsorokin@dk6n51 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
kvsorokin@dk6n51 ~/work/arch-pc/lab06 $ ./lab6-1
j
kvsorokin@dk6n51 ~/work/arch-pc/lab06 $

```

Рис. 4.3: Первая попытка выполнить

Уберём кавычки у числе в тексте программы(рис. 4.4).

```

7 mov eax,6
3 mov ebx,4

```

Рис. 4.4: Редактирование чисел

Попробуем ещё раз выполнить программу и увидим, что вывелся символ с кодом 10, а не 10(рис. 4.5).

```


kvsorokin@dk6n51 ~/work/arch-pc/lab06 $ gedit lab6-1.asm
kvsorokin@dk6n51 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
kvsorokin@dk6n51 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
kvsorokin@dk6n51 ~/work/arch-pc/lab06 $ ./lab6-1

```

Рис. 4.5: Вторая попытка выполнить

Создадим файл lab6-2.asm и введём в него более корректный код(рис. 4.6).


```
kvsorokin@dk6n51 ~/work/arch-pc/lab06 $ touch lab6-2.asm
kvsorokin@dk6n51 ~/work/arch-pc/lab06 $ ls
in_out.asm lab6-1 lab6-1.asm lab6-1.o lab6-2.asm
kvsorokin@dk6n51 ~/work/arch-pc/lab06 $ gedit lab6-2.asm
```



```
1 %include 'in_out.asm'
2 SECTION .text
3 GLOBAL _start
4 _start:
5 mov eax, '6'
6 mov ebx, '4'
7 add eax, ebx
8 call iprintLF
9 call quit
```

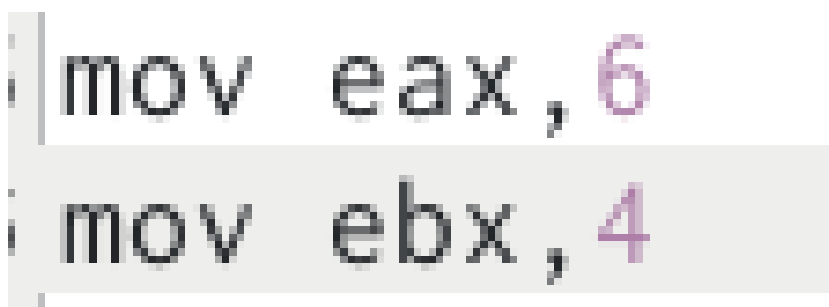
Рис. 4.6: Редактирование второго файла

После выполнения опять увидим, что вместо числа 10, пишется число 106(рис. 4.7).

```
kvsorokin@dk6n51 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
kvsorokin@dk6n51 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
kvsorokin@dk6n51 ~/work/arch-pc/lab06 $ ./lab6-2
bash: ./lab6-2: Нет такого файла или каталога
kvsorokin@dk6n51 ~/work/arch-pc/lab06 $ ./lab6-2
106
```

Рис. 4.7: Попытка запуска второго файла

Уберём кавычки у числе в тексте программы (рис. 4.8).



```
mov eax, 6
mov ebx, 4
```

Рис. 4.8: Изменение файла 2

Наконец после выполнения увидим желаемый ответ (рис. 4.9).

```

kvsorokin@dk6n51 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
kvsorokin@dk6n51 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
kvsorokin@dk6n51 ~/work/arch-pc/lab06 $ ./lab6-2
10

```

Рис. 4.9: Корректная работа программы

Создадим файл lab6-3.asm и запишем в него программу, выполняющую $(5 * 2 + 3)/3$ (рис. 4.10).

```

;-----
; 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 iprintfLF ; 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 iprintfLF ; из 'edx' (остаток) в виде символов
call quit ; вызов подпрограммы завершения

```

Рис. 4.10: Содержимое файла lab6-3.asm

Выполним программу и убедимся в верности полученного результата (рис. 4.11).

```

kvsorokin@dk6n51 ~/work/arch-pc/lab06 $ gedit lab6-3.asm
kvsorokin@dk6n51 ~/work/arch-pc/lab06 $ nasm -f elf lab6-3.asm
kvsorokin@dk6n51 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
kvsorokin@dk6n51 ~/work/arch-pc/lab06 $ ./lab6-3
Результат: 4
Остаток от деления: 1

```

Рис. 4.11: Выполнение программы вычисляющей $(5 * 2 + 3)/3$

Заменим значения в программе, чтобы она считала $(4*6 + 2)/5$ (рис. 4.12).

```

1 ; ---- Вычисление выраже
2 mov eax,4 ; EAX=5
3 mov ebx,6 ; EBX=2
4 mul ebx ; EAX=EAX*EBX
5 add eax,2 ; EAX=EAX+3
6 xor edx,edx ; обнуляем E
7 mov ebx,5 ; EBX=3
8 div ebx ; EAX=EAX/3, EDX:

```

Рис. 4.12: Подстановка других значений

Выполним у убедимся в верности выполнения (рис. 4.13).

```

kvsorokin@dk6n51 ~/work/arch-pc/lab06 $ gedit lab6-3.asm
kvsorokin@dk6n51 ~/work/arch-pc/lab06 $ nasm -f elf lab6-3.asm
kvsorokin@dk6n51 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
kvsorokin@dk6n51 ~/work/arch-pc/lab06 $ ./lab6-3
Результат: 5
Остаток от деления: 1

```

Рис. 4.13: Выполнение изменённой программы

Создадим файл variant.asm и скопируем туда текст программы называющей вариант задания (рис. ??).

```

;-----
; 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

```

Выполним, и узнаем, что по нашему билету получаем первый вариант (рис. 4.14).

```

kvsorokin@dk6n51 ~/work/arch-pc/lab06 $ touch ~/work/arch-pc/lab06/variant.asm
kvsorokin@dk6n51 ~/work/arch-pc/lab06 $ gedit variant.asm
kvsorokin@dk6n51 ~/work/arch-pc/lab06 $ nasm -f elf variant.asm
kvsorokin@dk6n51 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o variant variant.o
kvsorokin@dk6n51 ~/work/arch-pc/lab06 $ ./variant
Введите No студенческого билета:
1132236060
Ваш вариант: 1

```

Рис. 4.14: Мой вариант - 1

4.1 Ответы на вопросы:

1. 'mov eax, msg' и 'call sprintLF'
2. Для считывания и записывания информации в X с максимальной длиной 80
3. Для перевода введённых данных в число
4. Строки 'xor edx, edx', 'mov ebx, 20', 'div ebx', 'inc edx'
5. В edx
6. Для увеличения на значения на 1
7. 'mov eax, edx' и 'call iprintLF'

4.2 Самостоятельная работа

Создадим файл sam.asm для самостоятельной работы и напишем в нём программу согласно тому, что у нас первый вариант (рис. 4.15).

```
kvsorokin@dk6n51 ~/work/arch-pc/lab06 $ touch ~/work/arch-pc/lab06/sam.asm
kvsorokin@dk6n51 ~/work/arch-pc/lab06 $ gedit sam.asm
kvsorokin@dk6n51 ~/work/arch-pc/lab06 $ nasm -f elf sam.asm
kvsorokin@dk6n51 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o sam sam.o
kvsorokin@dk6n51 ~/work/arch-pc/lab06 $ ./sam
```

Рис. 4.15: Создание файла sam.asm

Основываясь на полученных знаниях напишем программу для вычисления выражения $(10+2x)/3$ (рис. 4.16).

```

#include 'in_out.asm' ; подключение внешнего файла
SECTION .data ; секция инициализированных данных
msg: DB 'Введите X: ',0
rem: DB 'Ответ: ',0
SECTION .bss ; секция не инициализированных данных
x: RESB 80 ; Переменная, значение к-рой будем вводить с
клавиатуры, выделенный размер - 80 байт
SECTION .text ; Код программы
GLOBAL _start ; Начало программы
_start: ; Точка входа в программу
; ---- Вычисление выражения
mov eax, msg ; запись адреса выводимого сообщения в eax
call sprint ; вызов подпрограммы печати сообщения
mov ecx, x ; запись адреса переменной в ecx
mov edx, 80 ; запись длины вводимого значения в edx
call sread ; вызов подпрограммы ввода сообщения
mov eax,x ; вызов подпрограммы преобразования
call atoi ; ASCII кода в число, eax=x

mov ebx,2
mul ebx
add eax,10
xor edx,edx ; обнуляем EDX для корректной работы div
mov ebx,3 ; EBX=3
div ebx

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

; ---- Вывод результата на экран
mov eax,rem ; вызов подпрограммы печати
call sprint ; сообщения 'Результат: '
mov eax,edi ; вызов подпрограммы печати значения
call iprintLF ; из 'edi' в виде символов
call quit

```

Рис. 4.16: Текст самостоятельной работы

Запустим программу и убедимся, что оба введенные значения выдают верный результат (рис. 4.17).

```
kvsorokin@dk6n51 ~/work/arch-pc/lab06 $ gedit sam.asm
kvsorokin@dk6n51 ~/work/arch-pc/lab06 $ nasm -f elf sam.asm
kvsorokin@dk6n51 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o sam sam.o
kvsorokin@dk6n51 ~/work/arch-pc/lab06 $ ./sam
Введите X: 1
Ответ: 4
kvsorokin@dk6n51 ~/work/arch-pc/lab06 $ ./sam
Введите X: 10
Ответ: 10
```

Рис. 4.17: Работа самостоятельной программы

5 Выводы

Мы научились использовать писать простые программы для вычисления на языке ассемблера

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://midnightcommander.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 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, 2017.
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. — БХВ- Петербург, 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,

17. — 1120 с. — (Классика Computer Science)