

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ № 7

дисциплина: *Архитектура компьютера*

Студент:

Гольденгорин Виталий Борисович

Группа:

НММ-01-2022

МОСКВА

2022 г.

Содержание

1	Цель работы...	4
2	Задание...	5
3	Теоретическое введение...	7
4	Выполнение лабораторной работы...	8
5	Выводы...	14

Список иллюстраций

Рис. 4.1...	8
Рис. 4.2...	8
Рис. 4.3...	8
Рис. 4.4...	9
Рис. 4.5...	9
Рис. 4.6...	9
Рис. 4.7...	9
Рис. 4.8...	10
Рис. 4.9...	10
Рис. 4.10...	11
Рис. 4.11...	11
Рис. 4.12...	12
Рис. 4.13...	13

1 Цель работы

Освоение арифметических инструкций языка ассемблера NASM.

2 Задание

1. Создайте каталог для программ лабораторной работы № 7, перейдите в него и создайте файл lab7-1.asm.
2. Введите в файл lab7-1.asm текст программы из листинга 7.1. Создайте исполняемый файл и запустите его.
3. Исправьте текст программы (Листинг 1). Создайте исполняемый файл и запустите его. Пользуясь таблицей ASCII определите какому символу соответствует код 10. Отображается ли этот символ при выводе на экран?
4. Преобразуем текст программы из Листинга 7.1 с использованием этих функций. Создайте файл lab7-2.asm в каталоге ~/work/arch-pc/lab07 и введите в него текст программы из листинга 7.2. Создайте исполняемый файл и запустите его.
5. Аналогично предыдущему примеру изменим символы на числа. Создайте исполняемый файл и запустите его. Какой результат будет получен при исполнении программы? Замените функцию `iprintLF` на `iprint`. Создайте исполняемый файл и запустите его. Чем отличается вывод функций `iprintLF` и `iprint`?
6. Создайте файл lab7-3.asm в каталоге ~/work/arch-pc/lab07. Внимательно изучите текст программы из листинга 7.3 и введите в lab7- 3.asm. Создайте исполняемый файл и запустите его. Измените текст программы для вычисления выражения $f(x) = (4 * 6 + 2)/5$. Создайте исполняемый файл и проверьте его работу.
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: $(Sn \bmod 20) + 1$, где Sn – номер студенческого билета (В данном случае $a \bmod b$ – это остаток от деления a на b).
 - вывести на экран номер варианта

Создайте файл variant.asm в каталоге ~/work/arch-pc/lab07. Внимательно изучите текст программы из листинга 7.4 и введите в файл variant.asm. Создайте исполняемый файл и запустите его. Проверьте результат работы программы вычислив номер варианта аналитическ

8. Включите в отчет по выполнению лабораторной работы ответы на следующие вопросы:
 1. Какие строки листинга 7.4 отвечают за вывод на экран сообщения ‘Ваш вариант:’?

2. Для чего используются следующие инструкции? `push mov ecx, x mov edx, 80 call sread`

3. Для чего используется инструкция “`call atoi`”?

4. Какие строки листинга 7.4 отвечают за вычисления варианта?

5. В какой регистр записывается остаток от деления при выполнении инструкции “`div ebx`”? 6. Для чего используется инструкция “`inc edx`”?

7. Какие строки листинга 7.4 отвечают за вывод на экран результата вычислений?

3 Теоретическое введение

- Регистровая адресация – операнды хранятся в регистрах и в команде используются имена этих регистров, например: `mov ax,bx`.
- Непосредственная адресация – значение операнда задается непосредственно в команде, Например: `mov ax,2`.
- Адресация памяти– операнд задает адрес в памяти. В команде указывается символическое обозначение ячейки памяти, над содержимым которой требуется выполнить операцию.
- Схема команды целочисленного сложения `add`
- Команда целочисленного вычитания `sub`
- `inc` (от англ. increment) и `dec` (от англ. decrement), которые увеличивают и уменьшают на 1 свой операнд.
- команда изменения знака `neg`
- Для беззнакового умножения используется команда `mul`
- Для знакового умножения используется команда `imul`
- Для деления, как и для умножения, существует 2 команды `div` и `idiv`:
- `iprint` – вывод на экран чисел в формате ASCII, перед вызовом `iprint` в регистр `eax` необходимо записать выводимое число
- `iprintLF` – работает аналогично `iprint`, но при выводе на экран после числа добавляет к символ перевода строки
- `atoi` – функция преобразует `ascii`-код символа в целое число и запишет результат в регистр `eax`, перед вызовом `atoi` в регистр `eax` необходимо записать число

4 Выполнение лабораторной работы

Сначала создадим файл lab7-1.asm в lab07.

```
[vitaliybg@fedora lab07]$ touch lab7-1.asm
```

Рис. 4.1: Создание lab7-1.asm

Потом я ввожу в файл lab7-1.asm текст программы из листинга 7.1 и создаю исполняемый файл, который я сразу же запускаю.



```
Открыть ▾ + lab7-1.asm  
~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab07  
  
%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. 4.2: Текст программы lab7-1.asm

```
[vitaliybg@fedora lab07]$ nasm -f elf lab7-1.asm  
[vitaliybg@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o  
[vitaliybg@fedora lab07]$ ./lab7-1  
j
```

Рис. 4.3: Создание исполняемого файла и его запуск

Исправляю текст программы, создаю исполняемый файл и запускаю его. В результате ничего не выводится.

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



```
[vitaliybg@fedora lab07]$ ld -m elf_i386 -o lab7-1-2 lab7-1-2.o
[vitaliybg@fedora lab07]$ ./lab7-1-2
```

Рис. 4.4: Изменение файла и его запуск

Создаю файл lab7-2.asm в lab07 и ввожу в него текст программы из листинга 7.2.



```
lab7-2.asm
~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab07

%include 'in_out.asm'

SECTION .text
GLOBAL _start
_start:
    mov eax, '6'
    mov ebx, '4'
    add eax, ebx
    call iprintLF

    call quit
```

Рис. 4.5: Текст программы lab7-2.asm

```
[vitaliybg@fedora lab07]$ touch lab7-2.asm
[vitaliybg@fedora lab07]$ nasm -f elf lab7-2.asm
[vitaliybg@fedora lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[vitaliybg@fedora lab07]$ ./lab7-2
106
```

Рис. 4.6: Создание файла lab7-2.asm и его запуск

Изменяю текст программы lab7-2.asm и запускаю его.



```
lab7-2.asm
~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab07

%include 'in_out.asm'

SECTION .text
GLOBAL _start
_start:
    mov eax, 6
    mov ebx, 4
    add eax, ebx
    call iprintLF

    call quit
```

```
[vitaliybg@fedora lab07]$ ld -m elf_i386 -o lab7-2-2 lab7-2.o
[vitaliybg@fedora lab07]$ ./lab7-2-2
106
[vitaliybg@fedora lab07]$
```

Рис. 4.8: Запуск измененной программы

Потом заменяю `iprintLF` на `iprint` и запускаю программу. Результат не поменялся.

```
call iprint
```

```
[vitaliybg@fedora lab07]$ ld -m elf_i386 -o lab7-2-3 lab7-2.o
[vitaliybg@fedora lab07]$ ./lab7-2-3
106
```

Рис. 4.9: Запуск программы с `iprint`

Создаю файл lab7-3.asm в lab07 и ввожу текст программы листинга 7.3 (вычисления выражения $f(x)=(5*2+3)/3$).



```
%include 'in_out.asm'

SECTION .data
div: DB 'Результат: ',0
rem: DB 'Остаток от деления: ',0

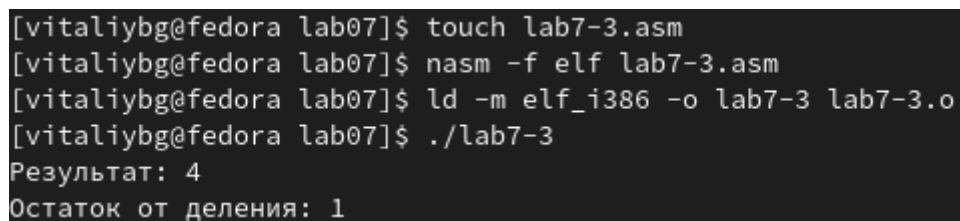
SECTION .text
GLOBAL _start
_start:
    mov eax,5
    mov ebx,2
    mul ebx
    add eax,3
    xor edx,edx
    mov ebx,3
    div ebx

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

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

    call quit
```

Рис. 4.10: Текст программы lab7-3.asm



```
[vitaliybg@fedora lab07]$ touch lab7-3.asm
[vitaliybg@fedora lab07]$ nasm -f elf lab7-3.asm
[vitaliybg@fedora lab07]$ ld -m elf_i386 -o lab7-3 lab7-3.o
[vitaliybg@fedora lab07]$ ./lab7-3
Результат: 4
Остаток от деления: 1
```

Рис. 4.11: Запуск исполняемого файла

Создаю файл variant.asm в lab07 и ввожу текст программы из листинга 7.4. Потом создаю исполняемый файл и запускаю его. Результат выводит 2 и совпадает с результатом алгоритма. За вывод на экране “Ваш вариант:” отвечают следующие строки команд:

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

Строки: mov ecx, x; mov edx, 80 call sread – отвечают за чтение введенного студенческого билета. Call atoi используется для приведения строки в числовой вид. За вычисление варианта отвечают строки: xor edx,edx mov ebx,20 div ebx inc edx. При выполнении div ebx остаток записывается в edx. Inc edx используется увеличить edx на 1. Строки: mov eax,rem call sprint mov eax,edx call iprintLF – отвечают за вывод на экран результат вычисления.

```
[vitaliybg@fedora lab07]$ touch variant.asm  
[vitaliybg@fedora lab07]$ nasm -f elf variant.asm  
[vitaliybg@fedora lab07]$ ld -m elf_i386 -o variant variant.o  
ld: невозможно найти variant.o: Нет такого файла или каталога  
[vitaliybg@fedora lab07]$ nasm -f elf variant.asm  
[vitaliybg@fedora lab07]$ ld -m elf_i386 -o vairant variant.o  
[vitaliybg@fedora lab07]$ ./variant  
bash: ./variant: Нет такого файла или каталога  
[vitaliybg@fedora lab07]$ ./vairant  
Введите No студенческого билета:  
1132226521  
Ваш вариант: 2  
[vitaliybg@fedora lab07]$
```

Рис. 4.12: Создание и запуск файла variant.asm

```

%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
    call atoi

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

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

    call quit

```

Рис. 4.13: Текст программы variant.asm

5 Выводы

После выполнения лабораторной работы я могу сделать следующий вывод: я научился работать с арифметикой на языке ассемблера NASM. Например, деление, умножение, прибавление и т.д.