

Отчёт по лабораторной работе %7

Дисциплина: 'архитектура компьютеров'

Бабенко Роман Игоревич

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выводы	18

Список иллюстраций

2.1	Создание каталога и файла	6
2.2	Записываем в файл	7
2.3	Запуск файла	8
2.4	Изменяем текст программы	9
2.5	Запуск изменённого файла	10
2.6	Создание второго файла	10
2.7	Создаем и запускаем исполняемый файл	10
2.8	Заменяем строки	11
2.9	Смотрим результат	12
2.10	Запуск изменённого файла	12
2.11	Создаём новый файл с программой	13
2.12	Запуск исполняемого файла и полученное результата	14
2.13	Результат выполнения изменённой программы	14
2.14	Получаем вариант %7	14
2.15	Получившаяся программа	16
2.16	Результаты тестов	17

Список таблиц

1 Цель работы

Освоить арифметические инструкции языка ассемблера NASM

2 Выполнение лабораторной работы

Создаём каталог для лабораторной работы %7 и файл 'lab7-1.asm' (рис. 2.1)

```
ribabenko@dk8n80 ~ $ mkdir ~/work/arch-pc/lab07
ribabenko@dk8n80 ~ $ cd ~/work/arch-pc/lab07
ribabenko@dk8n80 ~/work/arch-pc/lab07 $ touch lab7-1.asm
ribabenko@dk8n80 ~/work/arch-pc/lab07 $
```

Рис. 2.1: Создание каталога и файла

Записываем текст программы в файл (рис. 2.2)

```
%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. 2.2: Записываем в файл

Создаём исполняемый файл и запускаем его (рис. 2.3)

```
ribabenko@dk8n80 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
ribabenko@dk8n80 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
ribabenko@dk8n80 ~/work/arch-pc/lab07 $ ./lab7-1
j
ribabenko@dk8n80 ~/work/arch-pc/lab07 $ █
```

Рис. 2.3: Запуск файла

Изменим текст программы и вместо символов, запишем в регистры числа (рис. 2.4)


```
%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. 2.4: Изменяем текст программы

Создаём исполняемый файл и запускаем его. Символ с номером 10 - символ перевода строки (не отображается при выводе на экран)(рис. 2.5)

```
ribabenko@dk8n80 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
ribabenko@dk8n80 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
ribabenko@dk8n80 ~/work/arch-pc/lab07 $ ./lab7-1
ribabenko@dk8n80 ~/work/arch-pc/lab07 $
```

Рис. 2.5: Запуск изменённого файла

Создаём файл 'lab7-2' (рис. 2.6)

```
ribabenko@dk8n80 ~/work/arch-pc/lab07 $ touch ~/work/arch-pc/lab07/lab7-2.asm
ribabenko@dk8n80 ~/work/arch-pc/lab07 $
```

Рис. 2.6: Создание второго файла

Переписываем программу для вывода значения eax, создаём исполняемый файл и запускаем его (рис. 2.7)

```
ribabenko@dk8n80 ~/work/arch-pc/lab07 $ nasm -f elf lab7-2.asm
ribabenko@dk8n80 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-2 lab7-2.o
ribabenko@dk8n80 ~/work/arch-pc/lab07 $ ./lab7-2
106
```

Рис. 2.7: Создаем и запускаем исполняемый файл

Изменим символы на числа (рис. 2.8) и выполняем программу (рис. 2.9)

```
GNU nano 6.3 /afs/.d
#include 'in_out.asm'

SECTION .text
GLOBAL _start
_start:

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

call quit
```

Рис. 2.8: Заменим строки

```
GNU nano 6.3 /afs/.d
#include 'in_out.asm'

SECTION .text
GLOBAL _start
_start:

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

call quit
```

Рис. 2.9: Смотрим результат

Заменяем функцию `iprintLF` на `iprint` и запускаем файл (результат выводится на той же строке) (рис. 2.10)

```
ribabenko@dk8n80 ~/work/arch-pc/lab07 $ nasm -f elf lab7-2.asm
ribabenko@dk8n80 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-2 lab7-2.o
ribabenko@dk8n80 ~/work/arch-pc/lab07 $ ./lab7-2
10ribabenko@dk8n80 ~/work/arch-pc/lab07 $
```

Рис. 2.10: Запуск изменённого файла

Создаём файл `'lab7-3.asm'`, записываем в него предложенную программу (рис. 2.11)

```
...unk.sc1.pro.edu.i  
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. 2.11: Создаём новый файл с программой

Создаём исполняемый файл и запускаем его (рис. 2.12)

```
nasm -f elf lab7-3.asm
ribabenko@dk8n81 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-3 lab7-3.o
ribabenko@dk8n81 ~/work/arch-pc/lab07 $ ./lab7-3
Результат: 4
Остаток от деления: 1
ribabenko@dk8n81 ~/work/arch-pc/lab07 $
```

Рис. 2.12: Запуск исполняемого файла и полученное результата

Изменяем текст программы (изменяем выражение) и смотрим результат (рис. 2.13)

```
nasm -f elf lab7-3.asm
ribabenko@dk8n81 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-3 lab7-3.o
ribabenko@dk8n81 ~/work/arch-pc/lab07 $ ./lab7-3
Результат: 5
Остаток от деления: 1
ribabenko@dk8n81 ~/work/arch-pc/lab07 $
```

Рис. 2.13: Результат выполнения изменённой программы

Создаём новый файл, записываем в него программу для вычисления варианта по номеру студенческого билета (рис. 2.14)

```
ribabenko@dk8n80 ~/work/arch-pc/lab07 $ ./variant
Введите № студенческого билета:
1132226506
Ваш вариант: 7
ribabenko@dk8n80 ~/work/arch-pc/lab07 $
```

Рис. 2.14: Получаем вариант %7

#Ответы на вопросы

1. `mov eax,rem; call sprint`
2. Эти инструкции используются для ввода значения переменной с клавиатуры
3. Для преобразования кода ASCII в число
4. `xor edx,edx; mov ebx,20; div ebx; inc edx`

5. `edx`

6. Увеличивает значение на 1

7. `mov eax, edx; call iprintLF`

#Задание для самостоятельной работы

Напишем программу по 7 варианту, удовлетворяющую заданным условиям
(рис. 2.15)

```

SECTION .data

function: DB '5(x-1)^2',0
xznach: DB 'Введите x: ',0

answer: DB 'Результат: ',0

SECTION .bss
x: RESB 80

SECTION .text
GLOBAL _start
_start:

mov eax,function
call sprintf

mov eax, xznach
call sprintf

mov ecx, x
mov edx,80
call sread

mov eax,x
call atoi
add eax,-1
mul eax
mov edx,5
mul edx

mov edi,eax

mov eax,answer
call sprintf
mov eax,edi
call iprintf

call quit

```

Рис. 2.15: Получившаяся программа

Выполняем предложенные тесты и убеждаемся в корректности написанной программы (рис. 2.16)

```
ribabenko@dk8n80 ~/work/arch-pc/lab07 $ nasm -f elf task.asm
ribabenko@dk8n80 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o task task.o
ribabenko@dk8n80 ~/work/arch-pc/lab07 $ ./task
5(x-1)^2
Введите x: 3
Результат: 20
ribabenko@dk8n80 ~/work/arch-pc/lab07 $ ./task
5(x-1)^2
Введите x: 5
Результат: 80
ribabenko@dk8n80 ~/work/arch-pc/lab07 $
```

Рис. 2.16: Результаты тестов

3 Выводы

В ходе выполнения лабораторной работы я освоил арифметические инструкции языка ассемблера NASM