Отчет по лабораторной работе №6

Дисциплина: Архитектура компьютера

Мутаев Муртазаали Магомедович

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Освоение арифметических инструкция языка ассемблера NASM.

2 Задание

1. Листинг 1
2. Листинг 2
3. Листинг 3
4. Листинг 4
5. Задание для самостоятельной работы

3 Выполнение лабораторной работы

3.1 Листинг 1

Я создал файл lab6-1.asm в каталоге work/arch-pc/lab06, и в этот файл вставил код из Листинга 6.1 (рис. 3.1):

```
%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. 3.1: Листинг 6.1

Запустил программу и получил такой результат (рис. 3.2):

```
mmmutaev@dk6n62 ~/work/arch-pc/lab06 $ gedit lab6-1.asm
mmmutaev@dk6n62 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
mmmutaev@dk6n62 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
mmmutaev@dk6n62 ~/work/arch-pc/lab06 $ ./lab6-1
j
mmmutaev@dk6n62 ~/work/arch-pc/lab06 $
```

Рис. 3.2: Результат Листинга 6.1

Затем изменяю текст программы и вместо символов записываю в регистры числа и у меня выводится пустые символы, тк по таблице ASCII код 10 - пустной символ, наша программа вывела именно 10 код (рис. 3.3):

```
mmmutaev@dk6n62 ~/work/arch-pc/lab06 $ gedit lab6-1.asm
mmmutaev@dk6n62 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
mmmutaev@dk6n62 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
mmmutaev@dk6n62 ~/work/arch-pc/lab06 $ ./lab6-1


mmmutaev@dk6n62 ~/work/arch-pc/lab06 $
```

Рис. 3.3: Измененный Листинг 6.1

3.2 Листинг 2

Затем я создаю файл lab6-2.asm, ввожу текст из листинга и запускаю файл в работу: (рис. 3.4):

```
mmmutaev@dk6n62 ~/work/arch-pc/lab06 $ gedit lab6-2.asm
mmmutaev@dk6n62 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
mmmutaev@dk6n62 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
mmmutaev@dk6n62 ~/work/arch-pc/lab06 $ ./lab6-2
10
mmmutaev@dk6n62 ~/work/arch-pc/lab06 $
```

Рис. 3.4: Листинг 6.2

При исполнении этой программы я получил число 10, в этом мне помогла строчка iprintLF. Она выводит числа а не символы (рис. 3.5):

```
1 %include 'in_out.asm'
2
3 SECTION .text
4 GLOBAL _start
5  _start:
6
7  mov eax,'6'
8  mov ebx,'4'
9  add eax,ebx
10 call iprintLF
11
12 call quit
```

Рис. 3.5: Результат Листинга 6.2

Однако, если поменять текст программы и вместо iprintLF написать iprint, то у нас так же будет выведено число, но без переход на следующую строчку (рис. 3.6):

```
%include 'in_out.asm'

SECTION .text
GLOBAL _start
 _start:

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

 call quit
```

Рис. 3.6: Измененный Листинг 6.2

получится вот так: (рис. 3.7):

```
mmmutaev@dk3n33 ~/work/arch-pc/lab06 $ gedit lab6-2.asm
mmmutaev@dk3n33 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
mmmutaev@dk3n33 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
mmmutaev@dk3n33 ~/work/arch-pc/lab06 $ ./lab6-2
10mmmutaev@dk3n33 ~/work/arch-pc/lab06 $ ./lab6-2
10mmmutaev@dk3n33 ~/work/arch-pc/lab06 $
```

Рис. 3.7: Результат измененного Листинга 6.2

3.3 Листинг 3

теперь я ввожу программу из листинга 6.3 и запускаю: (рис. 3.8):

```
mmmutaev@dk3n33 ~/work/arch-pc/lab06 $ gedit lab6-3.asm
mmmutaev@dk3n33 ~/work/arch-pc/lab06 $ nasm -f elf lab6-3.asm
mmmutaev@dk3n33 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
mmmutaev@dk3n33 ~/work/arch-pc/lab06 $ ./lab6-3
Результат: 4
Остаток от деления: 1
mmmutaev@dk3n33 ~/work/arch-pc/lab06 $
```

Рис. 3.8: Результат Линстинга 6.3

А теперь изменяю текст программы для вычисления другого выражения следующим образом (рис. 3.9):

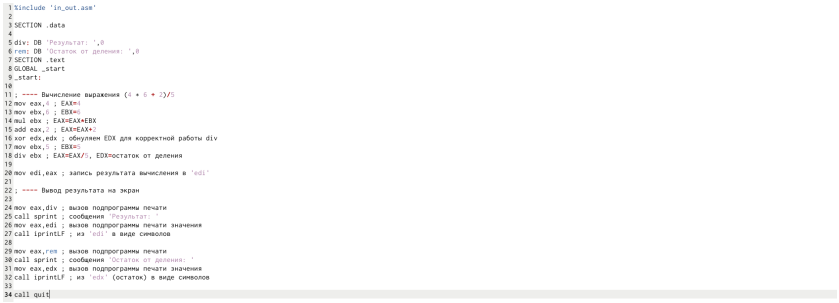

Рис. 3.9: Измененный Листинг 6.3

И программа выдала мне верный результат (рис. 3.10):

```
mmmutaev@dk8n60 ~/work/arch-pc/lab06 $ ./lab6-3
Результат: 5
Остаток от деления: 1
mmmutaev@dk8n60 ~/work/arch-pc/lab06 $
```

Рис. 3.10: Результат измененного Листинга 6.3

3.4 Листинг 4

В качестве другого примера рассмотрим программу вычисления варианта задания по номеру студенческого билета (рис. 3.11):

```
%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 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
```

Рис. 3.11: Вычисление варианта

Запустим программу и выясним мой вариант (рис. 3.12):

```
mmmutaev@dk8n60 ~/work/arch-pc/lab06 $ gedit variant.asm
mmmutaev@dk8n60 ~/work/arch-pc/lab06 $ nasm -f elf variant.asm
mmmutaev@dk8n60 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o variant variant.o
mmmutaev@dk8n60 ~/work/arch-pc/lab06 $ ./variant
Введите № студенческого билета:
1132246739
Ваш вариант: 20
mmmutaev@dk8n60 ~/work/arch-pc/lab06 $
```

Рис. 3.12: Вариант

Ответим на несколько вопросов:

1. Какие строки листинга 6.4 отвечают за вывод на экран сообщения ‘Ваш вариант:’?

За это отвечают строки 10-11

2. Для чего используется следующие инструкции? *mov ecx, x mov edx, 80 call sread*

Они используются для ввода и считывания значения переменной x

3. Для чего используется инструкция “call atoi”?

Чтобы преобразовать символьное значение в числовое

4. Какие строки листинга 6.4 отвечают за вычисления варианта?

За это отвечают строки 15-20

5. В какой регистр записывается остаток от деления при выполнении инструкции “div ebx”?

Он записывается в регистр edx

6. Для чего используется инструкция “inc edx”?

Увеличение значения регистра edx на 1

7. Какие строки листинга 6.4 отвечают за вывод на экран результата вычислений?

За это отвечают строки 23-24

##Задание для самостоятельной работы

Написать программу вычисления выражения y = f(x). Программа должна выводить выражение для вычисления, выводить запрос на ввод значения x, вычислять заданное выражение в зависимости от введенного x, выводить результат вычислений. Вид функции f(x) выбрать из таблицы 6.3 вариантов заданий в соответствии с номером полученным при выполнении лабораторной работы. Создайте исполняемый файл и проверьте его работу для значений x2 и x2 из 6.3

Для выполнения этого задани я написал следующую программу (рис. 3.13):

```
%include 'in_out.asm'

SECTION .data
msg: DB 'Выражение: x^3 * 1/3 + 21. Введите значение x: ', 0
rem: DB 'Ответ: ', 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

  mov ebx, eax
  mul ebx
  mul ebx
  xor edx, edx
  mov ebx, 3
  div ebx
  add eax, 21
  mov edi, eax

  mov eax, rem
  call sprint
  mov eax, edi
  call iprintLF

  call quit

```

Рис. 3.13: Задание для самостоятельной работы

У меня были вопросы в реализации умножения на 1/3, т.к. если записать это дробь в ассемблере, пользуясь знаниями из лекций, в лучшем случае получится

0, а в худшем - ошибка. Я мог умножить значение на 0, но думаю, что деление на 3 было более наглядным. Вот мой результат (рис. 3.14):

```
mmmutaev@dk8n60 ~/work/arch-pc/lab06 $ gedit stepik.asm
mmmutaev@dk8n60 ~/work/arch-pc/lab06 $ nasm -f elf stepik.asm
mmmutaev@dk8n60 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o stepik stepik.o
mmmutaev@dk8n60 ~/work/arch-pc/lab06 $ ./stepik
Выражение: x^3 * 1/3 + 21. Введите значение x:
1
Ответ: 21
mmmutaev@dk8n60 ~/work/arch-pc/lab06 $ ./stepik
Выражение: x^3 * 1/3 + 21. Введите значение x:
3
Ответ: 30
mmmutaev@dk8n60 ~/work/arch-pc/lab06 $
```

Рис. 3.14: Результат задания

4 Выводы

Я освоил арифметические инструкции языка ассемблера NASM.