Отчёт по лабораторной работе №6

Иванова Анастасия Сергеевна

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Освоить арифметические инструкции языка ассемблера NASM.

2 Порядок выполнения лабораторной работы

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

```
asivanova1@anastasia:~$ mkdir ~/work/arch-pc/lab06
asivanova1@anastasia:~$ cd ~/work/arch-pc/lab06
asivanova1@anastasia:~/work/arch-pc/lab06$ touch lab6-1.asm
```

Рисунок 2.1: Создание каталога и файла

2. Рассмотрим пример программ вывода символьных и численных значений. Программы будут выводить значения записанные в регистр eax. Введем в файл lab6-1.asm текст программы рис. 2.2:

```
 GNU nano 7.2     /home/asivanova1/work/arch-pc/lab06/lab6-1.asm
%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
```

Рисунок 2.2: Ввод программы

Создадим исполняемый файл и запустим его рис. 2.3:

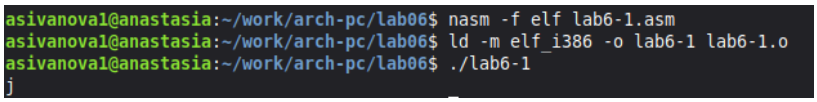

Рисунок 2.3: Проверка работы программы

В данном случае при выводе значения регистра eax мы ожидаем увидеть число 10. Однако результатом является символ j. Команда add eax,ebx записывает 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. Исправим текст программы рис. 2.4:

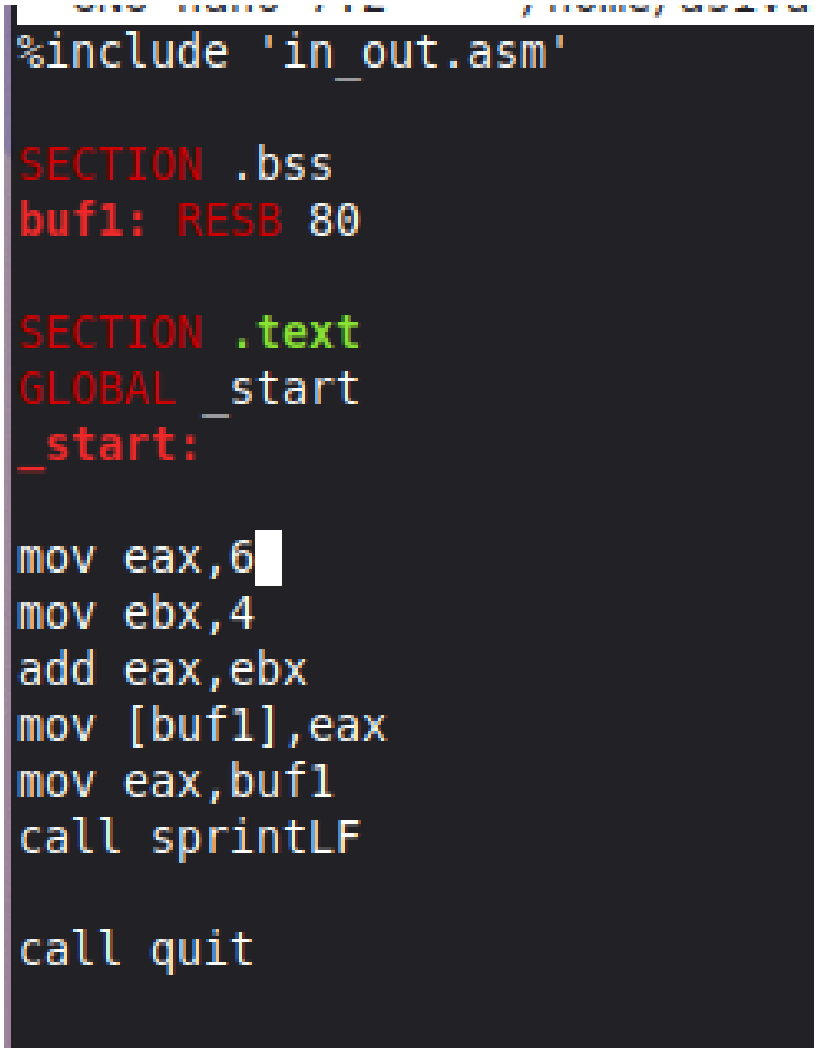

Рисунок 2.4: Изменение программы

Создадим исполняемый файл и запустим его рис. 2.5:

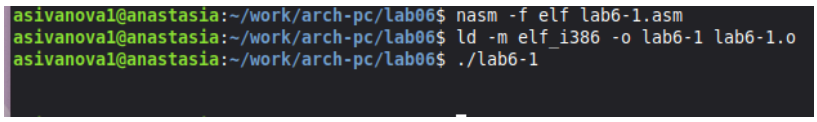

Рисунок 2.5: Проверка работы программы

Как и в предыдущем случае при исполнении программы мы не получим число 10. В данном случае должен выводится символ с кодом 10, но он не отобразился. 4. Для работы с числами в файле in_out.asm реализованы подпрограммы для преоб-

разования ASCII символов в числа и обратно. Создадим файл lab6-2.asm в каталоге ~/work/arch-pc/lab06 и введем в него текст программы рис. 2.6:

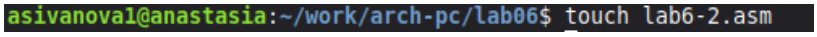

Рисунок 2.6: Создание файла

рис. 2.7

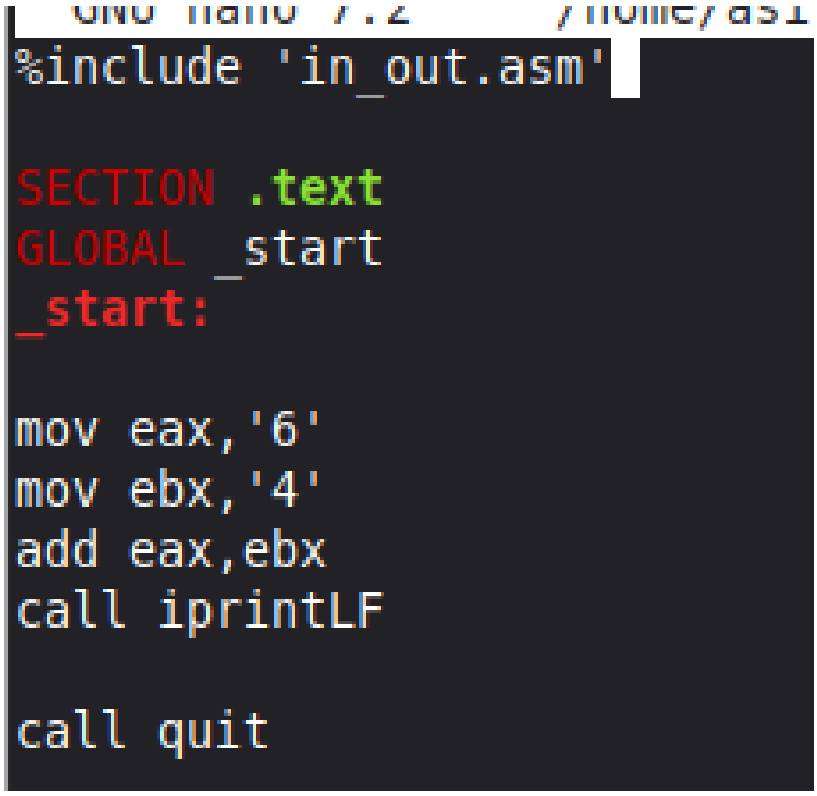

Рисунок 2.7: Ввод программы

Создадим исполняемый файл и запустим его рис. 2.8:

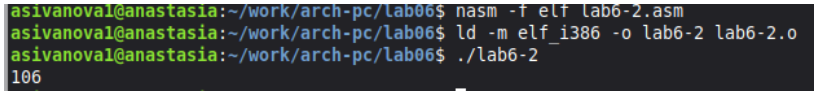

Рисунок 2.8: Проверка работы программы

В результате работы программы мы получаем число 106. 5. Аналогично предыдущему примеру изменим символы на числа. Заменим строки рис. 2.9:

```
%include 'in_out.asm'

SECTION .text
GLOBAL _start
_start:

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

call quit
```

Рисунок 2.9: Изменение программы

Создадим исполняемый файл и запустим его рис. 2.10:

```
asivanova1@anastasia:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
asivanova1@anastasia:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
asivanova1@anastasia:~/work/arch-pc/lab06$ ./lab6-2
10
```

Рисунок 2.10: Проверка работы программы

В результате работы программы мы получаем число 10. Заменим функцию iprintLF на iprint. Создадим исполняемый файл и запустим его рис. 2.11:

```
%include 'in_out.asm'

SECTION .text
GLOBAL _start
_start:

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

call quit
```

Рисунок 2.11: Изменение программы

рис. 2.12

```
asivanova1@anastasia:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
asivanova1@anastasia:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
asivanova1@anastasia:~/work/arch-pc/lab06$ ./lab6-2
10asivanova1@anastasia:~/work/arch-pc/lab06$
```

Рисунок 2.12: Запуск программы

После замены функции число 10 выводится без переноса (не на отдельной строчке). 6. В качестве примера выполнения арифметических операций в NASM приведем программу вычисления арифметического выражения f(x) = (5 * 2 + 3)/3. Создадим файл lab6-3.asm в каталоге ~/work/arch-pc/lab06 и напишем в нем программу рис. 2.13:

```
asivanova1@anastasia:~/work/arch-pc/lab06$ touch lab6-3.asm
```

Рисунок 2.13: Создание файла

рис. 2.14:

Рисунок 2.14: Ввод программы

Создадим исполняемый файл и запустим его рис. 2.15:

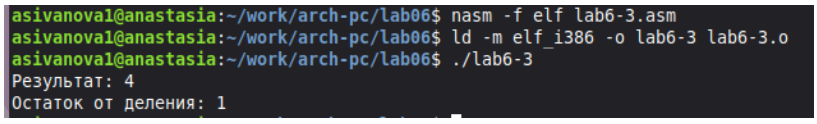

Рисунок 2.15: Проверка работы программы

Программа выводит результат - 4 и остаток от деления - 1. Изменим текст программы для вычисления выражения f(x) = (4 ∗ 6 + 2)/5 рис. 2.16:

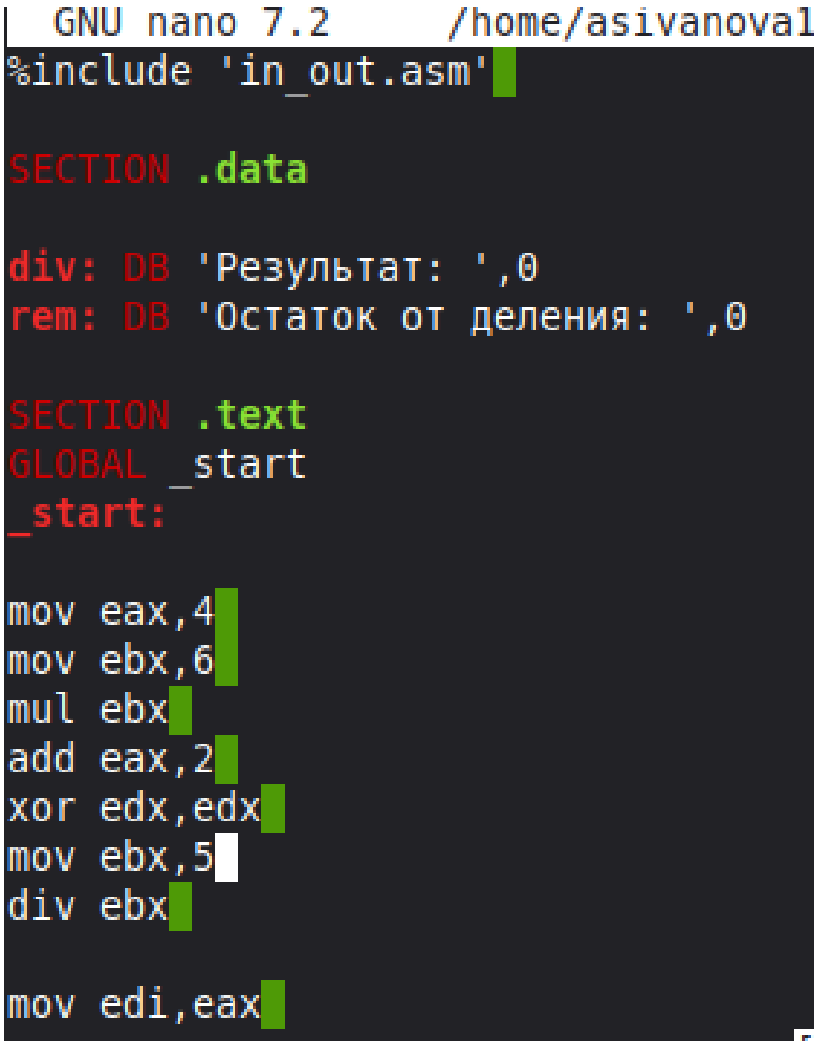

Рисунок 2.16: Изменение программы

Создадим исполняемый файл и проверим его работу рис. 2.17:

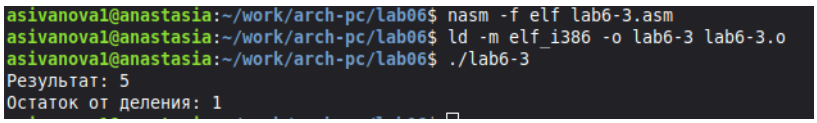

Рисунок 2.17: Проверка работы программы

Программа выводит результат - 5 и остаток от деления - 1. 7. В качестве другого примера рассмотрим программу вычисления варианта задания по номеру студенческого билета, работающую по следующему алгоритму: • вывести запрос на вве-

дение № студенческого билета • вычислить номер варианта по формуле: (Sn mod 20) + 1, где Sn – номер студенческого билета (В данном случае a mod b – это остаток от деления a на b). • вывести на экран номер варианта. Создадим файл variant.asm в каталоге ~/work/arch-pc/lab06 и напишем в нем программу рис. 2.18:

```
asivanova1@anastasia:~/work/arch-pc/lab06$ touch variant.asm
```

Рисунок 2.18: Создание файла

рис. 2.19

```
 GNU nano 7.2     /home/asivanova1/work/arch-pc/
%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
```

Рисунок 2.19: Ввод программы

Создадим исполняемый файл и запустим его рис. 2.20:

```
asivanova1@anastasia:~/work/arch-pc/lab06$ nasm -f elf variant.asm
asivanova1@anastasia:~/work/arch-pc/lab06$ ld -m elf_i386 -o variant variant.o
asivanova1@anastasia:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1132250427
Ваш вариант: 8
```

Рисунок 2.20: Проверка работы программы

Результатом работы является ответ - 8 вариант. 8. Ответы на следующие вопросы: 1. Какие строки листинга 6.4 отвечают за вывод на экран сообщения „Ваш вариант:“?: mov eax,rem call sprint 2. Для чего используется следующие инструкции? mov ecx, x - загрузка адреса буфера x в ECX mov edx, 80 - загрузка размера буфера EDX call sread - вызов функции чтения строки с клавиатуры 3. Для чего используется инструкция «call atoi»? - преобразование строки в число. 4. Какие строки листинга 6.4 отвечают за вычисления варианта?: xor edx,edx

mov ebx,20

div ebx

inc edx

5. В какой регистр записывается остаток от деления при выполнении инструкции «div ebx»? - в регистр EDX. 6. Для чего используется инструкция «inc edx»? - увеличение значения в регистре на 1. 7. Какие строки листинга 6.4 отвечают за вывод на экран результата вычислений?: mov eax,rem

call sprint

mov eax,edx

call iprintLF

3 Выполнение заданий для самостоятельной работы

1. Напишем программу для вычисления выражения y = (11 + x) * 2 - 6 рис. 3.1:

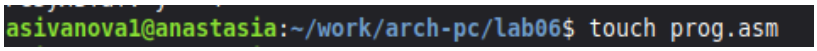

Рисунок 3.1: ВСоздание файла

рис. 3.2:

```
 GNU nano 7.2                  /home/asivanova1/work/arch-pc/lab06/prog.as
%include 'in_out.asm'

SECTION .data
    msg_yrav DB 'Вычисляем выражение: y = (11 + x)*2 - 6', 0
    msg_x DB 'Введите значение x: ', 0
    msg_y DB 'Результат: y = ', 0

SECTION .bss
x: RESB 80

SECTION .text
GLOBAL _start
_start:
mov eax, msg_yrav
call sprintLF

mov eax, msg_x
call sprint

mov ecx, x
mov edx, 10
call sread

mov eax, x
call atoi

add eax, 11
mov ebx, 2
mul ebx
sub eax, 6

mov edi, eax
mov eax, msg_y
call sprint
mov eax, edi
call iprintLF

call quit
```

Рисунок 3.2: Ввод программы

2. Создадим исполняемый файл и проверим его работу для значений x1 = 1 и x2 = 9 рис. 3.3:

```
asivanova1@anastasia:~/work/arch-pc/lab06$ nasm -f elf prog.asm
asivanova1@anastasia:~/work/arch-pc/lab06$ ld -m elf_i386 -o prog prog.o
asivanova1@anastasia:~/work/arch-pc/lab06$ ./prog
Вычисляем выражение: y = (11 + x)*2 - 6
Введите значение x: 1
Результат: y = 18
asivanova1@anastasia:~/work/arch-pc/lab06$ nasm -f elf prog.asm
asivanova1@anastasia:~/work/arch-pc/lab06$ ld -m elf_i386 -o prog prog.o
asivanova1@anastasia:~/work/arch-pc/lab06$ ./prog
Вычисляем выражение: y = (11 + x)*2 - 6
Введите значение x: 10
Результат: y = 36
```

Рисунок 3.3: Проверка работы программы

4 Вывод

Мы освоили арифметические инструкции языка ассемблера NASM.