РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 6

дисциплина: Архитектура компьютера

Студент: Ибрагимов Гаджимурад Шамильевич

Группа: НКАбд-02-25

МОСКВА

2025 г.

Содержание

Цель работы

1.1

Освоить арифметические операции относительно NASM Linux

Теоретическое введение

2.1. Адресация в NASM

Большинство инструкций на языке ассемблера требуют обработки операндов. Адрес операнда предоставляет место, где хранятся данные, подлежащие обработке. Это могут быть данные хранящиеся в регистре или в ячейке памяти. Далее рассмотрены все существующие способы задания адреса хранения операндов – способы адресации.

Существует три основных способа адресации:

• 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

2.2. Арифметические операции:

2.2.1 Целочисленное сложение add.

Схема команды целочисленного сложения add (от англ. addition - добавление) выполняет сложение двух операндов и записывает результат по адресу первогооперанда. Команда add

работает как с числами со знаком, так и без знака и выглядит следующим образом: add <операнд_1>, <операнд_2>

2.2.2. Целочисленное вычитание sub.

Команда целочисленного вычитания sub (от англ. subtraction – вычитание) работает аналогично команде add и выглядит следующим образом: sub <операнд_1>, <операнд_2>

2.2.3. Команды инкремента и декремента.

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. Эти команды содержат один операнд и имеет следующий вид:

inc <операнд>

dec <операнд>

2.2.4. Команда изменения знака операнда neg.

Еще одна команда, которую можно отнести к арифметическим командам это команда изменения знака neg: neg <операнд>

2.2.5.

Команды умножения mul и imul.

Умножение и деление, в отличии от сложения и вычитания, для знаковых и беззнаковых чисел производиться по-разному, поэтому существуют различные команды. Для беззнакового умножения используется команда mul (от англ. multiply – умножение):

mul <операнд>

Для знакового умножения используется команда imul:

imul <операнд>

2.2.6.

Команды деления div и idiv.

Для деления, как и для умножения, существует 2 команды div (от англ. divide - деление) и idiv:

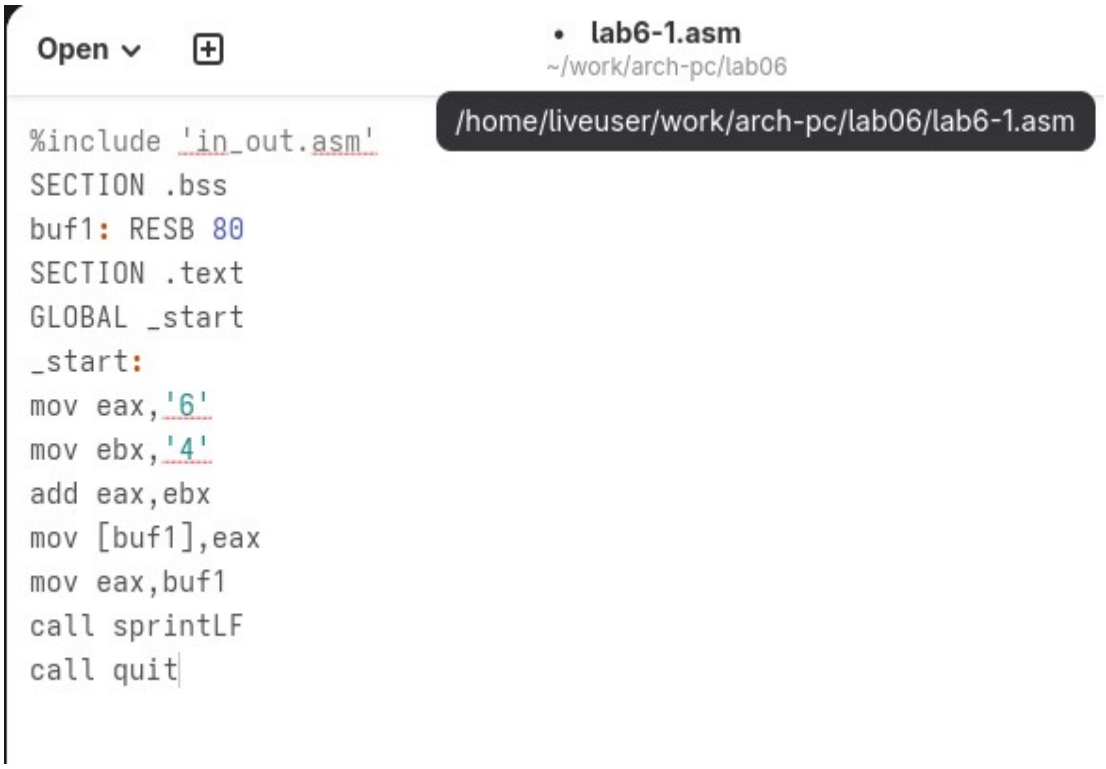
div <делитель> ; Беззнаковое деление

idiv <делитель> ; Знаковое деление

Выполнение лабораторной работы

3.1.1 Создаем каталог для программ лабораторной работы № 6, перейдите в него и создаем файл lab6-1.asm:

3.1.2. Рассмотрим примеры программ вывода символьных и численных значений. Программы будут выводить значения записанные в регистр eax

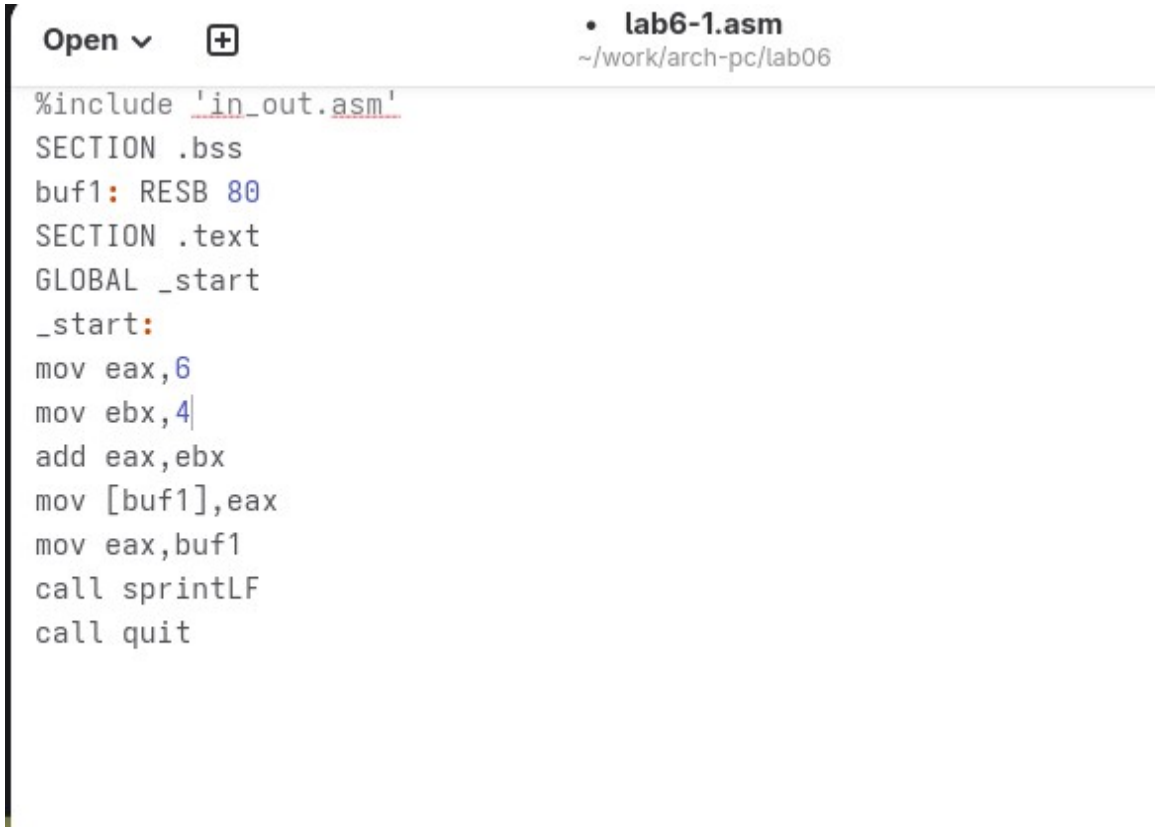


В данном случае при выводе значения регистра eax мы ожидаем увидеть число 10.

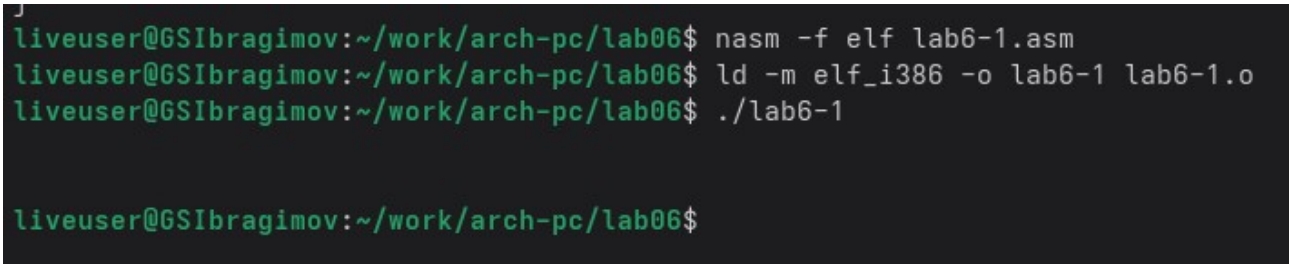
```
liveuser@GSIbragimov:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
liveuser@GSIbragimov:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
liveuser@GSIbragimov:~/work/arch-pc/lab06$ ./lab6-1
j
liveuser@GSIbragimov:~/work/arch-pc/lab06$
```

Однако результатом будет символ j. Это происходит потому, что код символа 6 равен 00110110 в двоичном представлении (или 54 в десятичном представлении), а код символа 4 – 0011010052). Команда add eax,ebx запишет в регистр eax сумму кодов – 01101010 (106), что в свою очередь является кодом символа j

3.1.3 Далее изменим текст программы и вместо символов, запишем в регистры числа. Исправьте текст программы следующим образом: заменим строки mov eax “6”, mov ebx, “4” на mov eax, 6   mov ebx, 4



Результатом будет символ перехода другую строку



3.2.1

Создайте файл lab6-2.asm в каталоге ~/work/arch-pc/lab06 и введите в него текст программы

```
liveuser@GSIbragimov:~/work/arch-pc/lab06$ touch lab6-2.asm
liveuser@GSIbragimov:~/work/arch-pc/lab06$
```

Текст программы с использованием функции iprintLF

В результате работы программы мы получим число 106. В данном случае, как и в первом, команда add складывает коды символов ‘6’ и ‘4’ (54+52=106). Однако, в отличии от программы из листинга функция iprintLF позволяет вывести число, а не символ, кодом которого является это число.

```
liveuser@GSIbragimov:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
liveuser@GSIbragimov:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
liveuser@GSIbragimov:~/work/arch-pc/lab06$ ./lab6-2
106
```

3.2.2

Аналогично предыдущему примеру изменим символы на числа.

Заменим строки mov eax “6”, mov ebx, “4” на mov eax, 6   mov ebx, 4

Результатом будет число 10, чего мы и ожидали от предыдущих программ. По сути происходит складывание двух чисел.

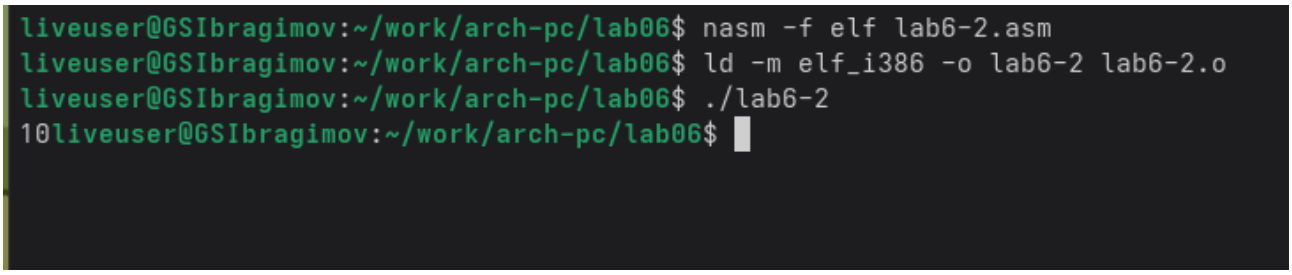
```
liveuser@GSIbragimov:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
liveuser@GSIbragimov:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
liveuser@GSIbragimov:~/work/arch-pc/lab06$ ./lab6-2
10
liveuser@GSIbragimov:~/work/arch-pc/lab06$
```

3.2.3

Заменим функцию iprintLF на iprint. Создадим исполняемый файл и

запум его. Чем отличается вывод функций iprintLF и iprint

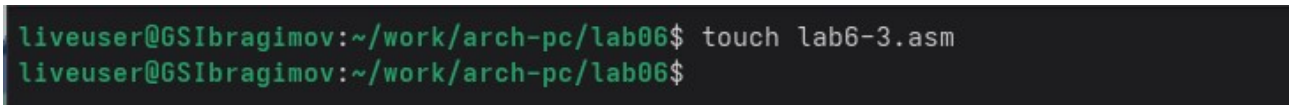
Довольно специфичный результат. Походит на оформление моей лабы)



3.3.1

В качестве примера выполнения арифметических операций в NASM приведем программу вычисления арифметического выражения $f(x) = (5 * 2 + 3)/3$.

Создаём файл lab6-3.asm в каталоге



Текст программы

Open • lab6-3.asm
~/work/arch-pc/lab06

```
%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 iprintLF ; 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 iprintLF ; из 'edx' (остаток) в виде символов
call quit ; вызов подпрограммы завершения
```

Создаем исполняемый файл и запускаем

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

3.3.2

Изменим текст программы для вычисления выражения

$f(x) = (4 * 6 + 2)/5$.

Создаем исполняемый файл и провем его работу.

Open • lab6-3.asm
~/work/arch-pc/lab06

```
%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,4 ; EAX=4
mov ebx,6 ; EBX=6
mul ebx ; EAX=EAX*EBX
add eax,2 ; EAX=EAX+2
xor edx,edx ; обнуляем EDX для корректной работы div
mov ebx,5 ; EBX=5
div ebx ; EAX=EAX/5, 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 iprintLF ; 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 iprintLF ; из 'edx' (остаток) в виде символов
call quit ; вызов подпрограммы завершения
```

По структуре идентичные выражения, поэтому переписываем значения для каждого регистра нашего алгоритма

Запускаем программу

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

3.4.1

В качестве другого примера рассмотрим программу вычисления варианта задания по номеру студенческого билета, работающую по следующему алгоритму:

• вывести запрос на введение № студенческого билета

Архитектура ЭВМ

• 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.

Создаем файл variant.asm в каталоге

```
liveuser@GSIbragimov:~/work/arch-pc/lab06$ touch variant.asm
liveuser@GSIbragimov:~/work/arch-pc/lab06$
```

Текст программы

Open • variant.asm
~/work/arch-pc/lab06

```
%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
```

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

Ответ: две строки mov eax, rem и call sprint

2. Для чего используется инструкции mov ecx, x…

Ответ: считывание номера студенческого билета

3. Для чего используется инструкция “call atoi”?

Ответ: вызов подпрограммы преобразования ASCII кода в число,

4. Какие строки листинга 6.4 отвечают за вычисления варианта?

Ответ:

xor edx,edx

mov ebx,20

div ebx

inc edx

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

Ответ: eax

6. Для чего используется инструкция “inc edx”?

Ответ: инкримент

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

Ответ:

mov eax,edx

call iprintLF

Результат работы программы

Выполнение самостоятельной работы

4.1

Мой вариант 13.

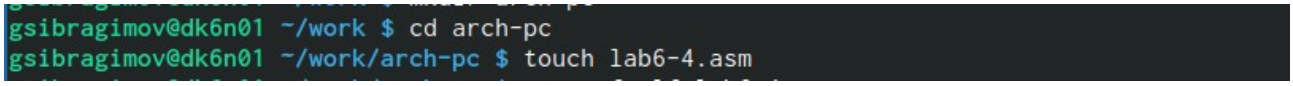
Следовательно:

Нужно написать программу вычисления выражения $(8x + 6) \cdot 10$.

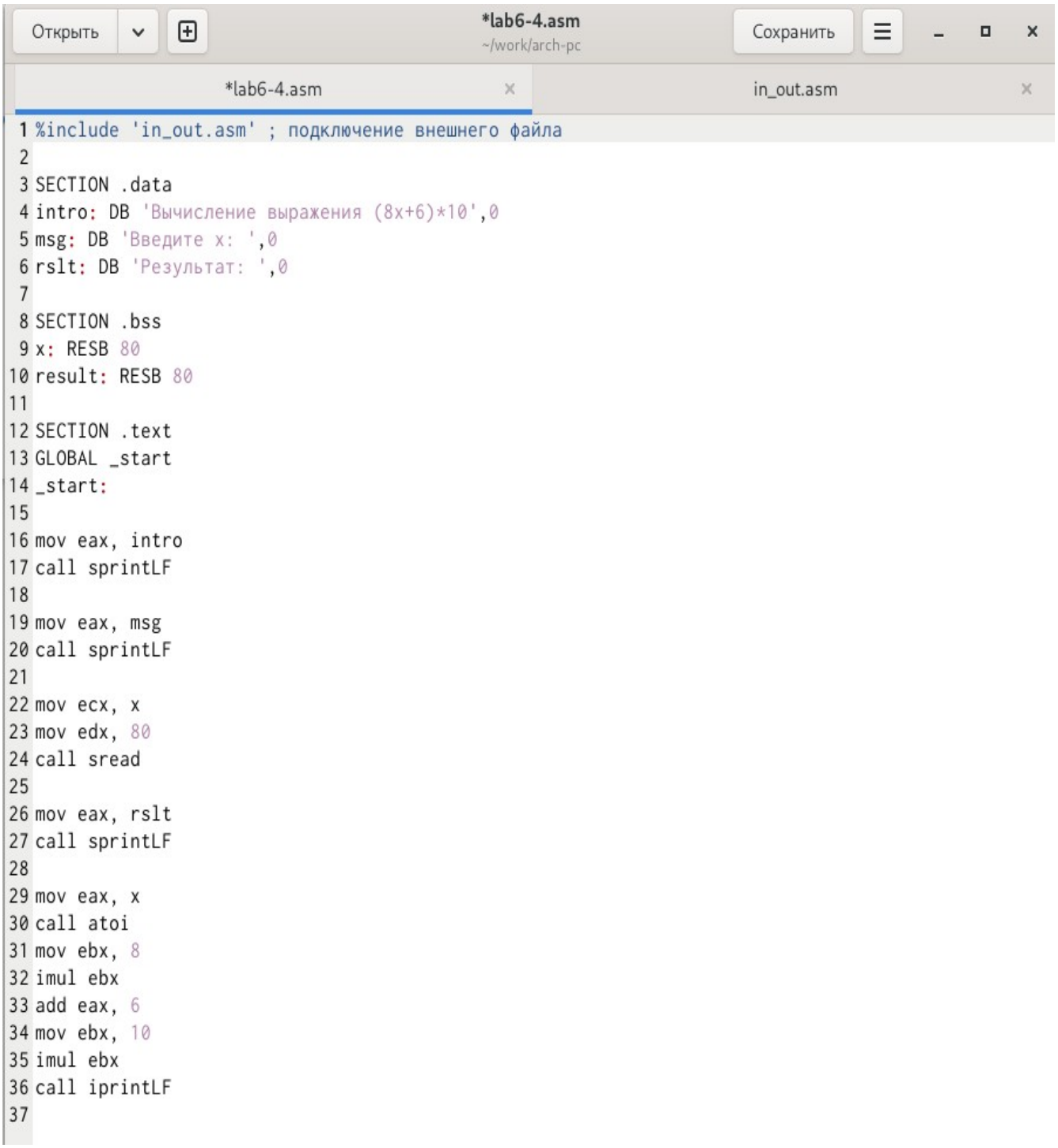
Программа должна выводить выражение для вычисления, выводить запрос на ввод значения x, вычислять заданное выражение в зависимости от введенного x, выводить результат вычислений. Создайте исполняемый файл и
проверьте его работу для значений 1 и 4.

П.С. Самостоятельную работу я проводил в дисплейном классе, а ход выполнения лабораторной работы на виртуальной машине дома.

Создаем файл с самостоятельной работой



Текст программы




```
1 %include 'in_out.asm' ; подключение внешнего файла
2
3 SECTION .data
4 intro: DB 'Вычисление выражения (8x+6)*10',0
5 msg: DB 'Введите x: ',0
6 rslt: DB 'Результат: ',0
7
8 SECTION .bss
9 x: RESB 80
10 result: RESB 80
11
12 SECTION .text
13 GLOBAL _start
14 _start:
15
16 mov eax, intro
17 call sprintLF
18
19 mov eax, msg
20 call sprintLF
21
22 mov ecx, x
23 mov edx, 80
24 call sread
25
26 mov eax, rslt
27 call sprintLF
28
29 mov eax, x
30 call atoi
31 mov ebx, 8
32 imul ebx
33 add eax, 6
34 mov ebx, 10
35 imul ebx
36 call iprintLF
37
```

По сути программа состоит из:

1. Строк вывода секции data
2. Строк считывания значения и заполнения значений секции bss (считывание в x)
3. Самого алгоритма вычисления выражения с 29 по 34 строки
4. Вывод ответа из регистра eax (от result пришлось отказаться)

Просмотрим различные результаты работы программы:

Для начала произведем сборку и запуск программы. После выполним операции от нас требуемые.

```
gsibragimov@dk6n01 ~/work/arch-pc $ nasm -f elf lab6-4.asm
gsibragimov@dk6n01 ~/work/arch-pc $ ld -m elf_i386 -o lab6-4  lab6-4.o
gsibragimov@dk6n01 ~/work/arch-pc $ ./lab6-4
Вычисление выражения (8x+6)*10
Введите x:
1
Результат:
140
```

При x = 1

```
gsibragimov@dk6n01 ~/work/arch-pc $ ./lab6-4
Вычисление выражения (8x+6)*10
Введите x:
4
Результат:
380
```

При x = 4

Для более больших чисел программа отрабатывает тоже

```
gsibragimov@dk6n01 ~/work/arch-pc $ ./lab6-4
Вычисление выражения (8x+6)*10
Введите x:
100
Результат:
8060
```

Вывод

В ходе выполнения данной лабораторной работы я познакомился с арифметическими операциями NASM.

Источники:

1. The NASM documentation. — 2021. — URL: https://www.nasm.us/docs.php

2. 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