Отчет по лабораторной работе №6

Дисциплина: архитектура компьютера

Абуков Ислам Ренатович

Цель работы

Цель данной лабораторной работы - освоение арифметческих инструкций языка ассемблера NASM.

Задание

1. Символьные и численные данные в NASM
2. Выполнение арифметических операций в NASM
3. Выполнение заданий для самостоятельной работы

Теоретическое введение

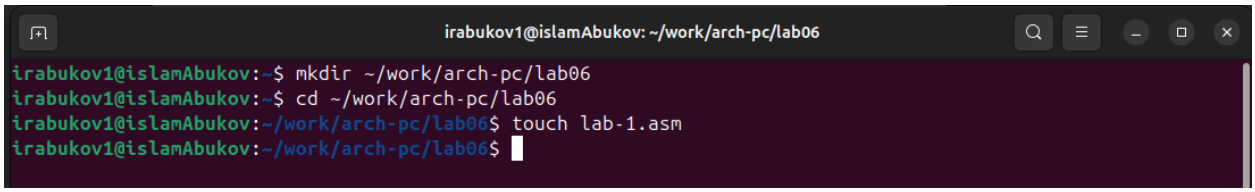
Большинство инструкций на языке ассемблера требуют обработки операндов. Адрес операнда предоставляет место, где хранятся данные, подлежащие обработке. Это могут быть данные хранящиеся в регистре или в ячейке памяти. - Регистровая адресация – операнды хранятся в регистрах и в команде используются имена 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. Ввод информации с клавиатуры и вывод её на экран осуществляется в символьном виде. Кодирование этой информации производится согласно кодовой таблице символов ASCII. ASCII – сокращение от American Standard Code for Information Interchange (Американский стандартный код для обмена информацией). Согласно стандарту ASCII каждый символ кодируется одним байтом. Среди инструкций NASM нет такой, которая выводит числа (не в символьном виде). Поэтому, например, чтобы вывести число, надо предварительно преобразовать его цифры в ASCII-коды этих цифр и выводить на экран эти коды, а не само число. Если же выводить число на экран непосредственно, то экран воспримет его не как число, а как последовательность ASCII-символов – каждый байт числа будет воспринят как один ASCII-символ – и выведет на экран эти символы. Аналогичная ситуация происходит и при вводе данных с клавиатуры. Введенные данные будут представлять собой символы, что сделает невозможным получение корректного результата при выполнении над ними арифметических операций. Для решения этой проблемы необходимо проводить преобразование ASCII символов в числа и обратно.

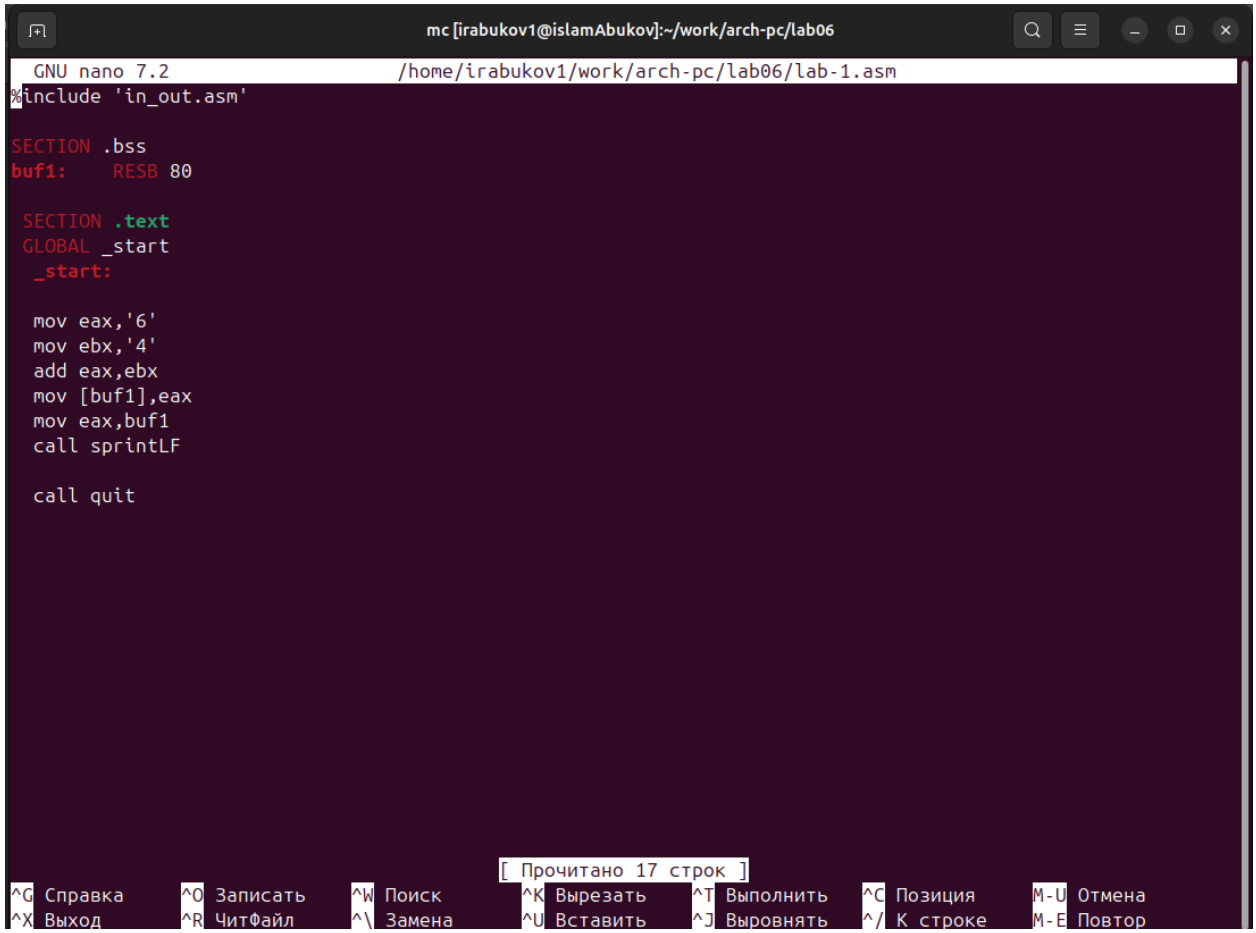
Выполнение лабораторной работы

Символьные и численные данные в NASM

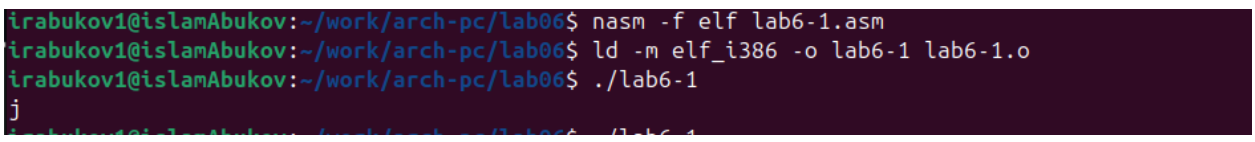
Создаю каталог для программ лабораторной работы №6 и перехожу в него, создаю там файл.



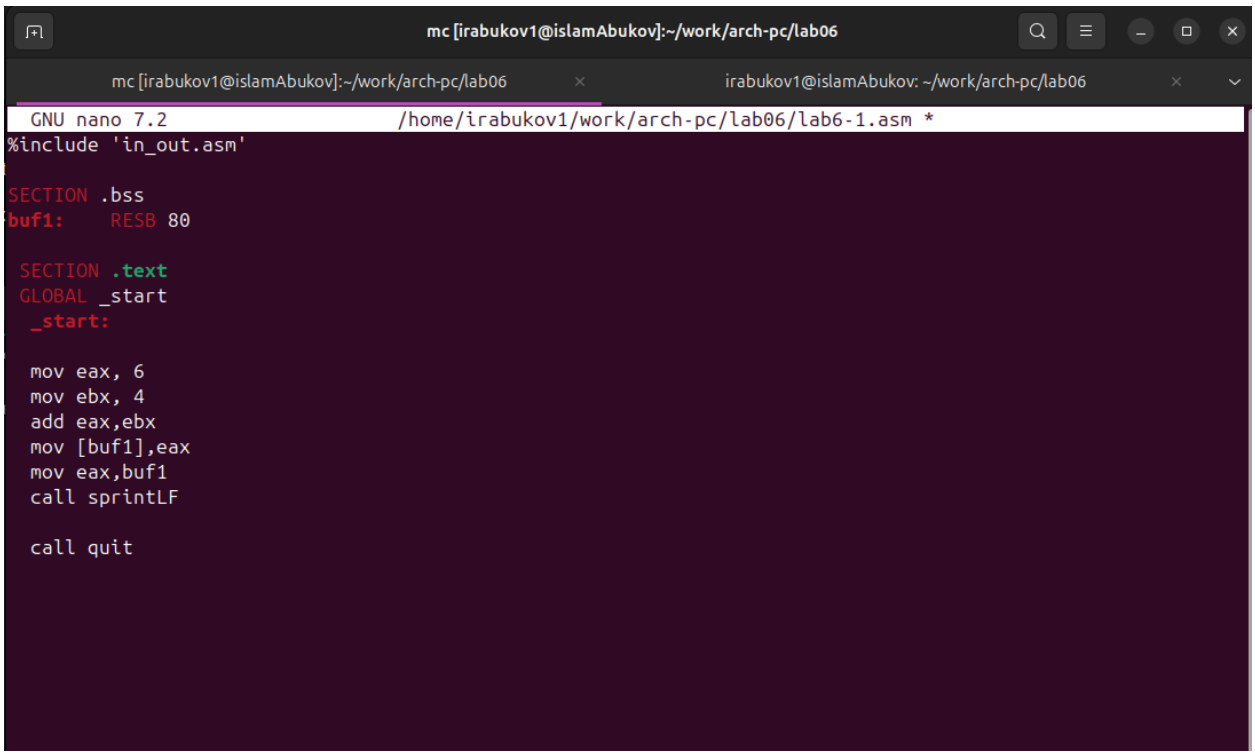
В созданном файле ввожу программу из листинга



Создаю исполняемый файл и запускаю его, вывод программы отличается от предполагаемого изначально, ибо коды символов в сумме дают символ j по таблице ASCII.



Изменяю текст изначальной программы, убрав кавычки.



На этот раз программа выдала пустую строчку, это связано с тем, что символ 10 означает переход на новую строку.

Создаю новый файл для будущей программы и записываю в нее код из листинга.

```
%include 'in_out.asm'

 SECTION .text
 GLOBAL _start
  _start:

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

  call quit
```

Создаю исполняемый файл и запускаю его, теперь отображается результат 106, программа, как и в первый раз, сложила коды символов, но вывела само число, а не его символ, благодаря замене функции вывода на iprintLF.

```
irabukov1@islamAbukov:~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/lab6-2.asm
irabukov1@islamAbukov:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
irabukov1@islamAbukov:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
irabukov1@islamAbukov:~/work/arch-pc/lab06$ ./lab6-2
106
irabukov1@islamAbukov:~/work/arch-pc/lab06$
```

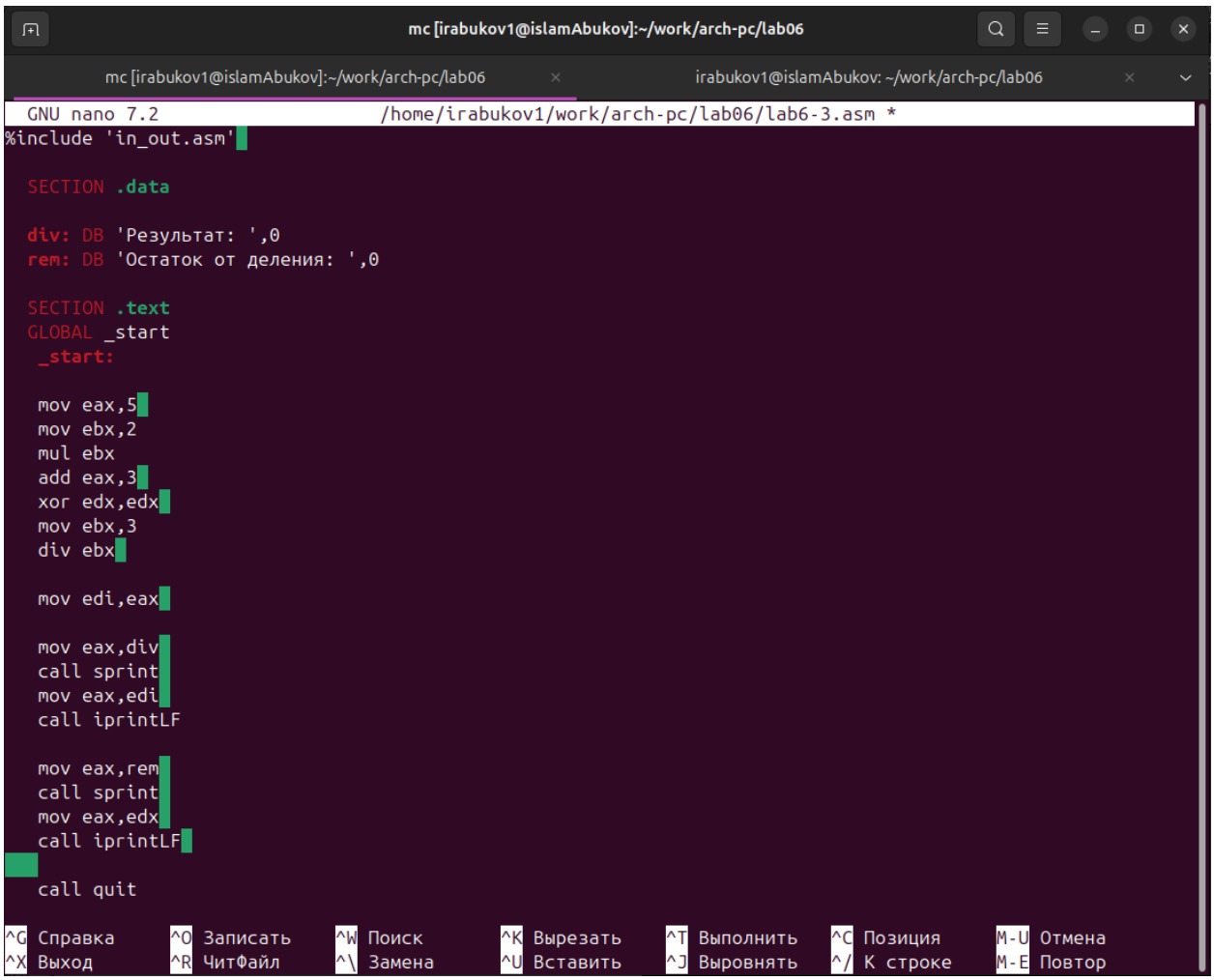
Убрав кавычки в программе, я снова ее запускаю и получаю предполагаемый изначально результат.

```
 GNU nano 7.2                  /home/irabukov1/work/arch-pc/lab06/lab6-2.asm *
%include 'in_out.asm'

 SECTION .text
 GLOBAL _start
  _start:

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

  call quit

Имя файла для записи: /home/irabukov1/work/arch-pc/lab06/lab6-2.asm
^G Справка          M-D Формат DOS      M-A Доп. в начало   M-B Резерв. копия
^C Отмена           M-M Формат Mac      M-P Доп. в конец    ^T Обзор
```

```
irabukov1@islamAbukov:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
irabukov1@islamAbukov:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-12 lab6-2.o
irabukov1@islamAbukov:~/work/arch-pc/lab06$ ./lab6-12
10
irabukov1@islamAbukov:~/work/arch-pc/lab06$
```

Выполнение арифметических операций в NASM

Создаю новый файл и копирую в него содержимое листинга.



Программа выполняет арифметические вычисления, на вывод идет результирующее выражения и его остаток от деления.

```
irabukov1@islamAbukov:~/work/arch-pc/lab06$ nasm -f elf lab6-3.asm
irabukov1@islamAbukov:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-3 lab6-3.o
irabukov1@islamAbukov:~/work/arch-pc/lab06$ ./lab6-3
Результат: 4
Остаток от деления: 1
irabukov1@islamAbukov:~/work/arch-pc/lab06$
```

Заменив переменные в программе для выражения f(x) = (4*6+2)/5.

```
 GNU nano 7.2                  /home/irabukov1/work/arch-pc/lab06/lab6-3.asm
%include 'in_out.asm'

  SECTION .data

  div: DB 'Результат: ',0
  rem: DB 'Остаток от деления: ',0

  SECTION .text
  GLOBAL _start
   _start:

   mov eax,4
   mov ebx,6
   mul ebx
   add eax,2
   xor edx,edx
   mov ebx,5
   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

                          [ Прочитано 32 строки ]
^G Справка     ^O Записать    ^W Поиск       ^K Вырезать    ^T Выполнить   ^C Позиция     M-U
^X Выход       ^R ЧитФайл     ^\ Замена      ^U Вставить    ^J Выровнять   ^/ К строке    M-E
```

Запуск программы дает корректный результат.

```
irabukov1@islamAbukov:~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/variant.asm
irabukov1@islamAbukov:~/work/arch-pc/lab06$ nasm -f elf lab6-3.asm
irabukov1@islamAbukov:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-31 lab6-3.o
irabukov1@islamAbukov:~/work/arch-pc/lab06$ ./lab6-31
Результат: 5
Остаток от деления: 1
irabukov1@islamAbukov:~/work/arch-pc/lab06$
```

Создаю новый файл и помещаю текст из листинга.

```
 GNU nano 7.2                  /home/irabukov1/work/arch-pc/lab06/variant.asm *
%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
  call atoi

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

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


^G Справка   ^O Записать  ^W Поиск     ^K Вырезать  ^T Выполнить  ^C Позиция   M-U Отмена
^X Выход     ^R ЧитФайл   ^\ Замена    ^U Вставить  ^J Выровнять  ^/ К строке  M-E Повтор
```

Запустив программу и указав свой номер студенческого билета, я получил свой вариант для дальнейшей работы.

```
irabukov1@islamAbukov:~/work/arch-pc/lab06$ nasm -f elf variant.asm
irabukov1@islamAbukov:~/work/arch-pc/lab06$ ld -m elf_i386 -o variant variant.o
irabukov1@islamAbukov:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1032251954
Ваш вариант: 15
irabukov1@islamAbukov:~/work/arch-pc/lab06$
```

Ответы на контрольные вопросы

1. За вывод сообщения “Ваш вариант” отвечают строки кода:

mov eax,rem

call sprint

2. Инструкция mov ecx, x используется, чтобы положить адрес вводимой строки x в регистр ecx mov edx, 80 - запись в регистр edx длины вводимой строки call sread - вызов подпрограммы из внешнего файла, обеспечивающей ввод сообщения с клавиатуры.

3. call atoi используется для вызова подпрограммы из внешнего файла, которая преобразует ascii-код символа в целое число и записывает результат в регистр eax.
4. За вычисления варианта отвечают строки:

```
xor edx,edx ; обнуление edx для корректной работы div mov

ebx,20 ; ebx = 20

div ebx ; eax = eax/20, edx - остаток от деления

inc edx ; edx = edx + 1
```

5. При выполнении инструкции div ebx остаток от деления записывается в регистр edx.
6. Инструкция inc edx увеличивает значение регистра edx на 1.
7. За вывод на экран результатов вычислений отвечают строки:

```
mov eax,edx

call iprintLF
```

Задание для самостоятельной работы

В соответствии с выбранным вариантом, я реализую программу для подсчета функции f(x) = (5 + x)^2 − 3, проверка на нескольких переменных показывает корректное выполнение программы.

```
 GNU nano 7.2                  /home/irabukov1/work/arch-pc/lab06/lab6-4.asm
%include 'in_out.asm'

 SECTION .data
 msg: DB 'Введите значение переменной x: ',0
 rem: DB 'Результат: ',0

 SECTION .bss
 x: RESB 80

 SECTION .text
 GLOBAL _start
  _start:

   mov eax, msg
   call sprint


   mov ecx, x
   mov edx, 80
   call sread


   mov eax, x
   call atoi
   mov ebx, eax


   add ebx, 5


   mov eax, ebx
   mul eax


                        [ Прочитано 46 строк ]
^G Справка    ^O Записать   ^W Поиск      ^K Вырезать   ^T Выполнить  ^C Позиция    M-U
```

Результат функции.

```
irabukov1@islamAbukov:~/work/arch-pc/lab06$ nasm -f elf lab6-3.asm
irabukov1@islamAbukov:~/work/arch-pc/lab06$ nasm -f elf lab6-4.asm
irabukov1@islamAbukov:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-4 lab6-4.o
irabukov1@islamAbukov:~/work/arch-pc/lab06$ ./lab6-4
Введите значение переменной x: 5
Результат: 97
irabukov1@islamAbukov:~/work/arch-pc/lab06$ ./lab6-4
Введите значение переменной x: 1
Результат: 33
```

Выводы

При выполнении данной лабораторной работы я освоил арифметические инструкции языка ассемблера NASM.