

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ № 7

дисциплина: Архитектура компьютера

Студент: Ужаков Магомед.

Группа: НПИбд-02-22

МОСКВА

2022 г.

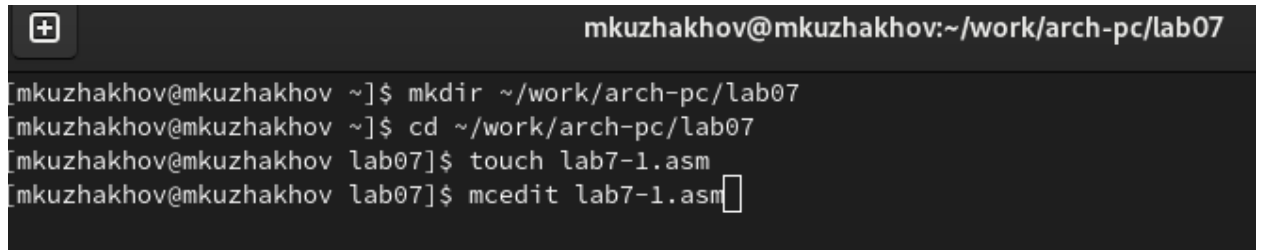
Цель работы:

Освоение арифметических инструкций языка ассемблера NASM.

Порядок выполнения лабораторной работы:

Символьные и численные данные в NASM.

Создадим каталог для программ лабораторной работы №7, перейдем в него и создадим файл lab7-1.asm (рис. 1).

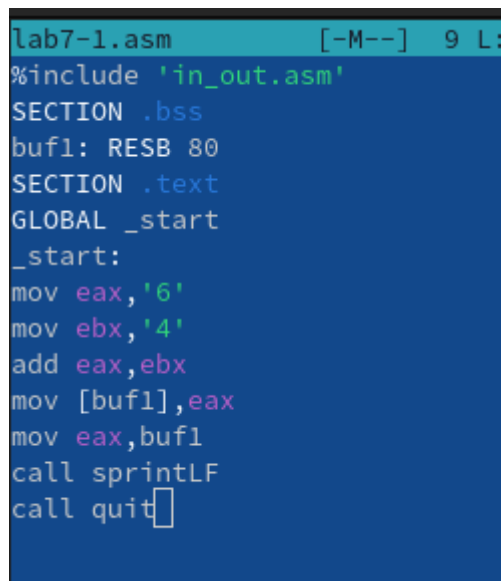


```
mkuzhakhov@mkuzhakhov:~/work/arch-pc/lab07
mkuzhakhov@mkuzhakhov ~]$ mkdir ~/work/arch-pc/lab07
mkuzhakhov@mkuzhakhov ~]$ cd ~/work/arch-pc/lab07
mkuzhakhov@mkuzhakhov lab07]$ touch lab7-1.asm
mkuzhakhov@mkuzhakhov lab07]$ mcedit lab7-1.asm
```

Рис. 1. Создание каталога и файла

Затем рассмотрим примеры программ вывода символьных и численных значений. Программы будут выводить значения, записанные в регистр `eax`.

Введем в файл `lab7-1.asm` текст программы (рис. 2). В данной программе в регистр `eax` записывается символ 6 (`mov eax,'6'`), в регистр `ebx` символ 4 (`mov ebx,'4'`). Далее к значению в регистре `eax` прибавляем значение регистра `ebx` (`add eax,ebx`, результат сложения запишется в регистр `eax`). Далее выводим результат. Так как для работы функции `sprintLF` в регистр `eax` должен быть записан адрес, необходимо использовать дополнительную переменную. Для этого запишем значение регистра `eax` в переменную `buf1` (`mov [buf1],eax`), а затем запишем адрес переменной `buf1` в регистр `eax` (`mov eax,buf1`) и вызовем функцию `sprintLF`.



```
lab7-1.asm [-M--] 9 L:
%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. 2. Код программы lab7-1

Затем создадим исполняемый файл и запустим его (рис. 3).

```
[mkuzhakhov@mkuzhakhov lab07]$ nasm -f elf lab7-1.asm
[mkuzhakhov@mkuzhakhov lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[mkuzhakhov@mkuzhakhov lab07]$ ./lab7-1
j
[mkuzhakhov@mkuzhakhov lab07]$
```

Рис. 3. Результат работы программы

В данном случае при выводе значения регистра `eax` мы ожидаем увидеть число 10. 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 – 00110100 (52). Команда `add eax,ebx` запишет в регистр `eax` сумму кодов – 01101010 (106), что в свою очередь является кодом символа `j` (см. таблицу ASCII).

Далее изменим текст программы и вместо символов, запишем в регистры числа (рис. 4).

```
_start:
mov  eax,6
mov  ebx,4
add  eax,ebx
mov  [buf1],eax
```

Рис. 4. Исправленный код программы lab7-1

Создадим исполняемый файл и запустим его (рис. 5).

```
j
[mkuzhakhov@mkuzhakhov lab07]$ mcedit lab7-1.asm

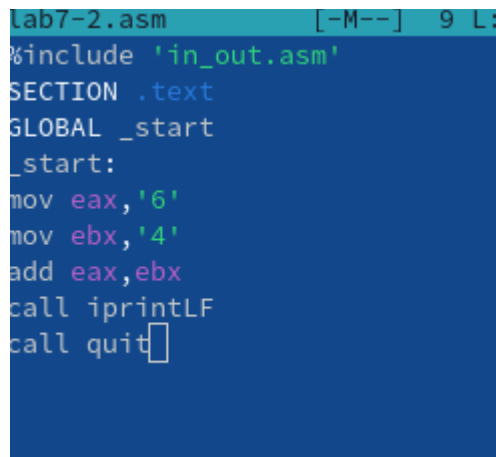
[mkuzhakhov@mkuzhakhov lab07]$ nasm -f elf lab7-1.asm
[mkuzhakhov@mkuzhakhov lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[mkuzhakhov@mkuzhakhov lab07]$ ./lab7-1

[mkuzhakhov@mkuzhakhov lab07]$
```

Рис. 5. Результат работы программы

Как и в предыдущем случае при исполнении программы мы не получим число 10, в данном случае выводится символ с кодом 10, этому коду соответствует управляющий символ перевода строки.

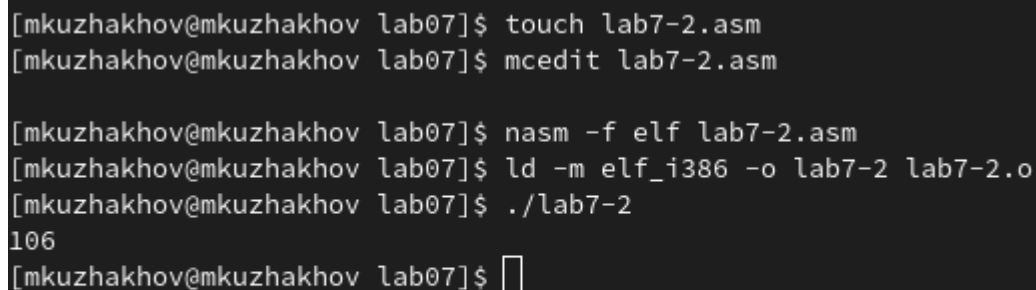
Для работы с числами в файле `in_out.asm` реализованы подпрограммы для преобразования ASCII символов в числа и обратно. Преобразуем текст программы с использованием этих функций. Создадим файл `lab7-2.asm` и введем в него следующий текст (рис. 6).



```
lab7-2.asm [-M--] 9 L:
#include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,'6'
mov ebx,'4'
add eax,ebx
call iprintLF
call quit
```

Рис. 6. Код программы lab7-2

Создадим исполняемый файл и запустим его (рис. 7).



```
[mkuzhakhov@mkuzhakhov lab07]$ touch lab7-2.asm
[mkuzhakhov@mkuzhakhov lab07]$ mcedit lab7-2.asm

[mkuzhakhov@mkuzhakhov lab07]$ nasm -f elf lab7-2.asm
[mkuzhakhov@mkuzhakhov lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[mkuzhakhov@mkuzhakhov lab07]$ ./lab7-2
106
[mkuzhakhov@mkuzhakhov lab07]$
```

Рис. . Результат работы программы

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 отличие от программы из листинга 7.1, функция `iprintLF` позволяет вывести число, а не символ, кодом которого является это число.

Аналогично предыдущему примеру изменим символы на числа (рис. 8).

```
lab7-2.asm [-M--] 9
#include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,6
mov ebx,4
add eax,ebx
call iprintLF
call quit
```

Рис. 7. Изменение кода программы lab7-2

В итоге при выполнении программы получился следующий результат (рис. 9).

```
остаток от деления: 0
[mkuzhakhov@mkuzhakhov lab07]$ nasm -f elf lab7-2.asm
[mkuzhakhov@mkuzhakhov lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[mkuzhakhov@mkuzhakhov lab07]$ ./lab7-2
10
[mkuzhakhov@mkuzhakhov lab07]$
```

Рис. 8.

Результат работы программы lab7-2

Заменяем функцию iprintLF из рис. 8 на iprint (рис. 10).

```
[mkuzhakhov@mkuzhakhov lab07]$ nasm -f elf lab7-2.asm
[mkuzhakhov@mkuzhakhov lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[mkuzhakhov@mkuzhakhov lab07]$ ./lab7-2
10[mkuzhakhov@mkuzhakhov lab07]$
```

Рис. 9. Результат работы программы с iprint

Как видим, отличие команды iprint от iprintLF заключается в том, что команда iprint не переводит строку.

Выполнение арифметических операций в NASM.

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

Создадим файл lab7-3.asm в каталоге ~/work/arch-pc/lab07 и введем в него следующий текст (рис. 11).

```
mkuzhakhov@mkuzhakhov:~/work/arch-p
lab7-3.asm [----] 41 L:[ 1+25 26/ 26] *(1236/1236b) <E0
#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
```

Рис. 10. Код программы lab7-3

Создадим исполняемый файл и запустим программу (рис. 12).

```

[mkuzhakhov@mkuzhakhov lab07]$ mcedit lab7-3.asm

[mkuzhakhov@mkuzhakhov lab07]$ nasm -f elf lab7-3.asm
[mkuzhakhov@mkuzhakhov lab07]$ ld -m elf_i386 -o lab7-3 lab7-3.o
[mkuzhakhov@mkuzhakhov lab07]$ ./lab7-3
Результат: 4
Остаток от деления: 1
[mkuzhakhov@mkuzhakhov lab07]$ 

```

Рис. 11. Результат работы программы lab7-3

Изменим текст программы для вычисления выражения $f(x) = (4 * 6 + 2)/5$ (рис. 13). Затем создадим файл и проверим его работу (рис. 14).

```

; ---- Вычисление выражения
mov eax,4 ; EAX=5
mov ebx,6 ; EBX=2
mul ebx ; EAX=EAX*EBX
add eax,2 ; EAX=EAX+3
xor edx,edx ; обнуляем EDX
mov ebx,5 ; EBX=3
div ebx ; EAX=EAX/3, EDX=остаток
mov edi,eax ; запись результата

```

Рис. 12. Код новой программы lab7-3

```

[mkuzhakhov@mkuzhakhov lab07]$ mcedit lab7-3.asm

[mkuzhakhov@mkuzhakhov lab07]$ nasm -f elf lab7-3.asm
[mkuzhakhov@mkuzhakhov lab07]$ ld -m elf_i386 -o lab7-3 lab7-3.o
[mkuzhakhov@mkuzhakhov lab07]$ ./lab7-3
Результат: 5
Остаток от деления: 1
[mkuzhakhov@mkuzhakhov lab07]$ 

```

Рис. 13. Результат работы программы lab7-3

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

- вывести запрос на введение № студенческого билета

- вычислить номер варианта по формуле: $(Sn \bmod 20) + 1$, где Sn – номер студенческого билета (В данном случае $a \bmod b$ – это остаток от деления a на b).
- вывести на экран номер варианта.

В данном случае число, над которым необходимо проводить арифметические операции, вводится с клавиатуры. Как отмечалось выше ввод с клавиатуры осуществляется в символьном виде и для корректной работы арифметических операций в NASM символы необходимо преобразовать в числа. Для этого используется функция `atoi` из файла `in_out.asm`.

Создадим файл `variant.asm` в каталоге `~/work/arch-pc/lab07` и напишем в нем код программы (рис. 15).

```
variant.asm      [-M--]  9 L:[ 1+24 25/ 25
%include 'in_out.asm'
SECTION .data
msg: DB 'Введите No 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. 14. Код программы variant

```
[mkuzhakhov@mkuzhakhov lab07]$ mcedit variant.asm

[mkuzhakhov@mkuzhakhov lab07]$ nasm -f elf variant.asm
[mkuzhakhov@mkuzhakhov lab07]$ ld -m elf_i386 -o variant variant.o
[mkuzhakhov@mkuzhakhov lab07]$ ./variant
Введите No студенческого билета:
1132222846
Ваш вариант: 7
[mkuzhakhov@mkuzhakhov lab07]$
```

Рис. 15. Результат работы программы variant

Как видим, мой вариант для следующего задания №15.

Ответы на вопросы:

1. За вывод на экран сообщения 'Ваш вариант: ' отвечают следующие строки:
rem: DB 'Ваш вариант: ',0
mov eax,rem
call sprint
2. Инструкция nasm используется для преобразования текста программы в объектный код; инструкция mov esx,x используется для записи адреса под вводимую строку; инструкция mov edx, 80 используется для определения длины вводимой строки; инструкция call sread используется для ввода сообщения с клавиатуры.
3. Инструкция call atoi используется для преобразования ascii-кода символа в целое число и записывает результат в регистр eax.
4. За вычисление варианта отвечают следующие строки кода:
mov eax,x
call atoi
xor edx,edx
mov ebx,20
div ebx
inc edx
5. Остаток от деления при выполнении инструкции div ebx записывается в регистр edx.
6. Инструкция inc edx используется для увеличения значения регистра edx на 1.

7. За вывод на экран результата вычислений отвечают следующие строки кода:

```
mov eax,edx  
call iprintLF
```

Порядок выполнения самостоятельной работы:

Напишем программу вычисления выражения, в соответствии с вариантом, полученным в предыдущем задании - вариант № 7. Моё выражение будет следующим: $5(x-1)^2$. Создадим файл sr1.asm и напомним код (рис. 17).



```
sr1.asm [----] 9 L: [ 1+30  
%include 'in_out.asm'  
SECTION .data  
msg DB 'Enter x...', 0  
ans DB 'Answer = ', 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 eax, 3  
dec eax  
mul eax  
mov ebx, 5  
mul ebx  
  
mov ebx, eax  
mov eax, ans  
call sprint  
mov eax, ebx  
call iprintLF  
  
call quit
```

Рис. 16. Код программы sr1

Затем создадим исполняемый файл, запустим программу и проверим его для значений $x_1 = 3$; $x_2 = 5$ (рис. 18).

```
[mkuzhakhov@mkuzhakhov lab07]$ nasm -f elf sr1.asm
[mkuzhakhov@mkuzhakhov lab07]$ ld -m elf_i386 -o sr1 sr1.o
[mkuzhakhov@mkuzhakhov lab07]$ ./sr1
Enter x...
3
Answer = 20
```

Рис. 17. Результат работы программы sr1

Выводы:

Во время выполнения лабораторной работы были освоены арифметические инструкции языка ассемблера NASM: add – сложение, sub – вычитание, mul – умножение, div – деление нацело, inc – увеличение на 1, dec – уменьшение на 1, neg – изменение знака числа.

