Отчет по лабораторной работе №6

Арифметические операции в NASM

ФИО: **Маркеш Виейра Нанке Грасимилде**

1 Цель работы

Освоение арифметических инструкций языка ассемблера NASM

2 Задание

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

3 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.

4 Выполнение лабораторной работы

4.1 Символьные и численные данные в NASM

С помощью утилиты mkdir создаю директорию, в которой буду создавать файлы с программами для лабораторной работы №6 (рис. 4.1). Перехожу в созданный каталог с помощью утилиты cd. С помощью утилиты touch создаю файл lab6-1.asm и с помощью утилиты ls проверяю, что файл действительно был создан (рис. 4.1).

Рис. 4.1: Создание директории

Копирую в текущий каталог файл in_out.asm с помощью утилиты cp, т.к. он будет использоваться в других программах (рис. 4.2).

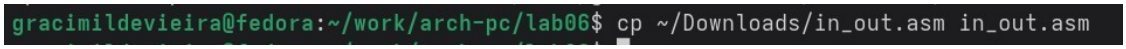

Рис. 4.2: Создание копии файла

Открываю созданный файл lab6-1.asm, вставляю в него программу вывода значения регистра eax (рис. 4.3).

Рис. 4.3: едактирование файла

Создаю исполняемый файл программы и запускаю его (рис. 4.4). Вывод программы: символ j, потому что программа вывела символ, соответствующий по системе ASCII сумме двоичных кодов символов 4 и 6.

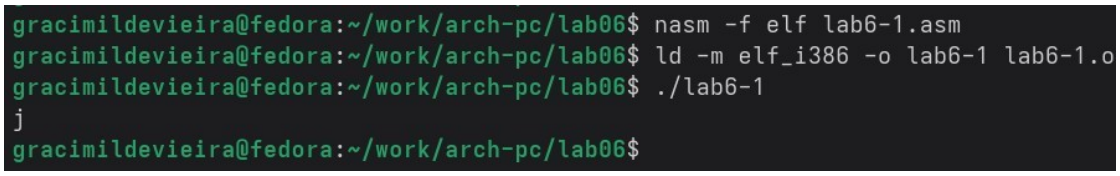

Рис. 4.4: Запуск исполияемого файла

Изменяю в тексте программы символы “6” и “4” на цифры 6 и 4 (рис. 4.5).

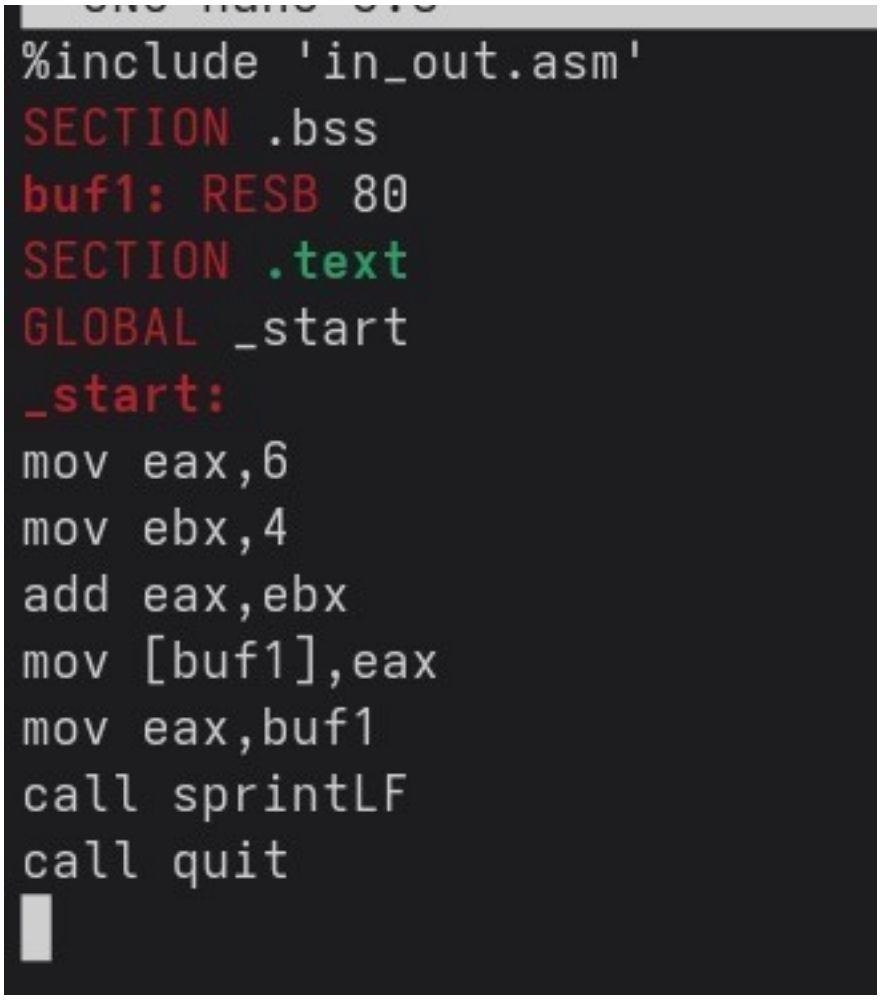

Рис. 4.5: Редактирование файла

Создаю новый исполняемый файл программы и запускаю его (рис. 4.6). Теперь вывелся символ с кодом 10, это символ перевода строки, этот символ не отображается при выводе на экран.

Рис. 4.6: Запуск исполияемого файла

Создаю новый файл lab6-2.asm с помощью утилиты touch и проверяю с помощью ls, что файл был создан (рис. 4.7).

```
gracimildevieira@fedora:~/work/arch-pc/lab06$ touch lab6-2.asm
gracimildevieira@fedora:~/work/arch-pc/lab06$ ls
in_out.asm  lab6-1  lab6-1.asm  lab6-1.o  lab6-2.asm
gracimildevieira@fedora:~/work/arch-pc/lab06$
```

Рис. 4.7 :Создание файла

Ввожу в файл текст другой программы для вывода значения регистра eax (рис. 4.8).

```
%include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,'6'
mov ebx,'4'
add eax,ebx
call iprintLF
call quit
```

Рис. 4.8: Редактирование файла

Создаю и запускаю исполняемый файл lab6-2 (рис. 4.9). Теперь вывод число 106, потому что программа позволяет вывести именно число, а не символ, хотя все еще происходит именно сложение кодов символов “6” и “4”.

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

Рис. 4.9: Запуск исполияемого файла

Заменяю в тексте программы в файле lab6-2.asm символы “6” и “4” на числа 6 и 4 (рис. 4.10).

```
%include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,6
mov ebx,4
add eax,ebx
call iprintLF
call quit
```

Рис. 4.10: Редактирование файла

Создаю и запускаю новый исполняемый файл (рис. 4.11).. Теперь программа складывает не соответствующие символам коды в системе ASCII, а сами числа, поэтому вывод 10.

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

Рис. 4.11: Запуск исполияемого файла

Заменяю в тексте программы функцию iprintLF на iprint (рис. 4.12).

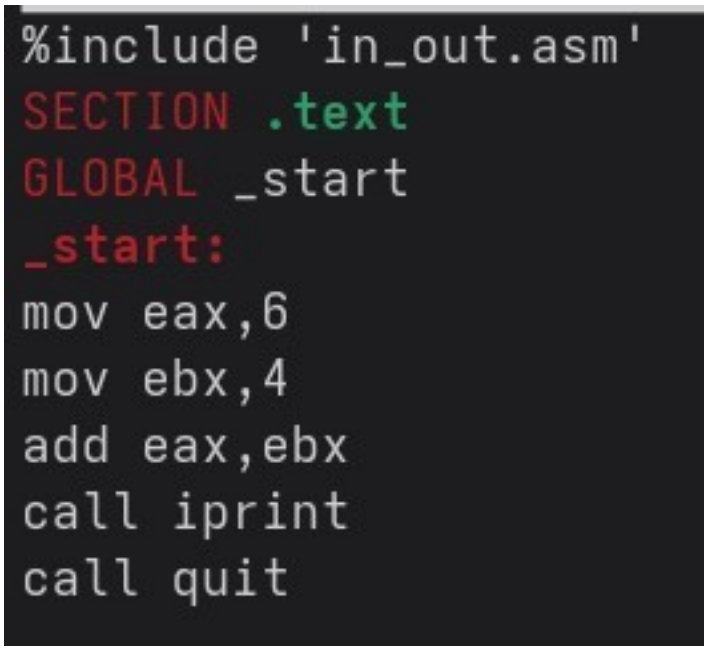

Рис. 4.12: Редактирование файла

Создаю и запускаю новый исполняемый файл (рис. 4.13). Вывод не изменился, потому что символ переноса строки не отображался, когда программа исполнялась с функцией iprintLF, а iprint не добавляет к выводу символ переноса строки, в отличие от iprintLF.

```
gracimildevieira@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
gracimildevieira@fedora:~/work/arch-pc/lab06$ ./lab6-2
10gracimildevieira@fedora:~/work/arch-pc/lab06$
```

Рис. 4.13: Запуск исполняемого файла

4.2 Выполнение арифметических операций в NASM

Создаю файл lab6-3.asm с помощью утилиты touch (рис. 4.14).

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

Рис. 4.14: Создание файла

Ввожу в созданный файл текст программы для вычисления значения выражения f**(x) = (5 * 2 + 3)/3** (рис. 4.15).

Abrir

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

```
1 %include 'in_out.asm' ; подключение внешнего файла
2 SECTION .data
3 div: DB 'Результат: ',0
4 rem: DB 'Остаток от деления: ',0
5 SECTION .text
6 GLOBAL _start
7 _start:
8 ; ---- Вычисление выражения
9 mov eax,5 ; EAX=5
10 mov ebx,2 ; EBX=2
11 mul ebx ; EAX=EAX*EBX
12 add eax,3 ; EAX=EAX+3
13 xor edx,edx ; обнуляем EDX для корректной работы div
14 mov ebx,3 ; EBX=3
15 div ebx ; EAX=EAX/3, EDX=остаток от деления
16 mov edi,eax ; запись результата вычисления в 'edi'
17 ; ---- Вывод результата на экран
18 mov eax,div ; вызов подпрограммы печати
19 call sprint ; сообщения 'Результат: '
20 mov eax,edi ; вызов подпрограммы печати значения
21 call iprintLF ; из 'edi' в виде символов
22 mov eax,rem ; вызов подпрограммы печати
23 call sprint ; сообщения 'Остаток от деления: '
24 mov eax,edx ; вызов подпрограммы печати значения
25 call iprintLF ; из 'edx' (остаток) в виде символов
26 call quit ; вызов подпрограммы завершения
```

Рис. 4.15: Редактирование файла

Создаю исполняемый файл и запускаю его (рис. 4.16).

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

Рис. 4.16: Запуск исполняемого файла

Изменяю программу так, чтобы она вычисляла значение выражения **f(x) = (4 * 6 + 2)/5** (рис. 4.17).

Abrir

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

```
1 %include 'in_out.asm' ; подключение внешнего файла
2 SECTION .data
3 div: DB 'Результат: ',0
4 rem: DB 'Остаток от деления: ',0
5 SECTION .text
6 GLOBAL _start
7 _start:
8 ; ---- Вычисление выражения
9 mov eax,4 ; EAX=4
10 mov ebx,6 ; EBX=6
11 mul ebx ; EAX=EAX*EBX
12 add eax,2 ; EAX=EAX+2
13 xor edx,edx ; обнуляем EDX для корректной работы div
14 mov ebx,5 ; EBX=5
15 div ebx ; EAX=EAX/5, EDX=остаток от деления
16 mov edi,eax ; запись результата вычисления в 'edi'
17 ; ---- Вывод результата на экран
18 mov eax,div ; вызов подпрограммы печати
19 call sprint ; сообщения 'Результат: '
20 mov eax,edi ; вызов подпрограммы печати значения
21 call iprintLF ; из 'edi' в виде символов
22 mov eax,rem ; вызов подпрограммы печати
23 call sprint ; сообщения 'Остаток от деления: '
24 mov eax,edx ; вызов подпрограммы печати значения
25 call iprintLF ; из 'edx' (остаток) в виде символов
26 call quit ; вызов подпрограммы завершения
```

Рис. 4.17: Измеиение программы

Создаю и запускаю новый исполняемый файл (рис. 4.18). Я посчитала для проверки правильности работы программы значение выражения самостоятельно, программа отработала верно.

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

Рис. 4.18: Запуск исполияемого файла

Создаю файл variant.asm с помощью утилиты touch (рис. 4.19).

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

Рис. 4.19: Редактирование файла

Ввожу в файл текст программы для вычисления варианта задания по номеру студенческого билета (рис. 4.20).

Abrir *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 пребразования
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. 4.20: Редактирование файла

Создаю и запускаю исполняемый файл (рис. 4.21). Ввожу номер своего студ. билета с клавиатуры, программа вывела, что мой вариант - 17.

```
gracimildevieira@fedora:~/work/arch-pc/lab06$ nasm -f elf variant.asm
gracimildevieira@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o variant variant.o
gracimildevieira@fedora:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1032255356
Ваш вариант: 17
gracimildevieira@fedora:~/work/arch-pc/lab06$
```

Рис. 4.21: Запуск исполняемого файла

4.3 Вопросы по программе

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
```

это запись в регистр edx длины вводимой строки

```
call sread
```

это 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
```

4.4 Задание для самостоятельной работы

Создаю файл lab6-4.asm с помощью утилиты touch (рис. 4.22).

```
gracimildevieira@fedora:~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/lab6-4.asm
gracimildevieira@fedora:~/work/arch-pc/lab06$
```

Рис. 4.22: Создание файла

Открываю созданный файл для редактирования, ввожу в него текст программы для вычисления значения выражения 18(x + 1)/6 (рис. 4.23). Это выражение было под вариантом 17.

```
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data
msg: DB 'Введите значение переменной x: ',0
rem: DB 'Результат: ',0
SECTION .bss
x: RESB  80; переменная, значение которой будем вводить с клавиатуры
SECTION .text
GLOBAL _start
_start:
; ---- Вычисление выражения
mov eax, msg
call sprint
mov eax, x
mov edx, 80
call sread
mov eax,x ; вызов подпрограммы Преобразования
call atoi ; ASCII кода в число, 'eax=x'
add eax, 1;   eax = eax 1 = x + 1
mov ebx,eax
imul eax,ebx; EAX = 18 * (x + 1)
mov ebx, 6
imul eax,ebx; EAX = 18(x + 1) / 6
mov edi,eax ; запись результата вычисления в 'edi'
; ---- Вывод результата на экран
mov eax,rem ; вызов подпрограммы печати
call sprint ; сообщения 'результат: '
mov eax,edi ; вызов подпрограммы печати значения
call iprint ; из 'edi' в виде символов
call quit ; вызов подпрограммы завершения
```

Рис. 4.23: Написание программы

Рис. **4.23**: Написание программы

Создаю и запускаю исполняемый файл (рис. 4.24). При вводе значения 3, вывод - 108.

```
gracimildevieira@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-4.asm
gracimildevieira@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-4 lab6-4.o
gracimildevieira@fedora:~/work/arch-pc/lab06$ ./lab6-4
Введите значение переменной x: 3
Результат: 108gracimildevieira@fedora:~/work/arch-pc/lab06$
```

Рис. 4.24: Запуск исполняемого файла

Провожу еще один запуск исполняемого файла для проверки работы программы с другим значением на входе - 1 (рис. 4.25). Программа отработала верно и в ответе мы получили 6.

```
gracimildevieira@fedora:~/work/arch-pc/lab06$ ./lab6-4
Введите значение переменной x: 1
Результат: 6gracimildevieira@fedora:~/work/arch-pc/lab06$
```

Рис. 4.25: Запуск исполняемого файла

Листинг **4.1.** Программа для вычисления значения выражения **18(x + 1)/6.**

```
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data
msg: DB 'Введите значение переменной x:',0
rem: DB 'Результат: ',0
SECTION .bss
x: RESB 80; Перемеиная, значение которой будем вводить с клавиатуры
SECTION .text
GLOBAL _start
_start:
;-----Вычисление выражения
mov eax,msg
call sprint
mov ecx, x
mov edx, 80
call sread
mov eax,x ; вызов подпрограммы преобразрвания
call atoi ; ASCII кода в число, `eax=x`
add eax, 1; eax = eax 1 = x + 1
mov ebx,eax
imul eax,ebx; EAX = 18 * (x +1)
mov ebx,6
imul eax,ebx; EAX = 18(x+1)/6
mov edi,eax ; запись результата вычисления в 'edi'
;-----Вывод результата на экран
mov eax,rem ; вызов подпрограммы печати
```

```
call sprint ; сообщения 'Результат: '
mov eax,edi ; вызов подпрограммы печати значения
call iprint ; из 'edi' в виде символов
call quit ; вызов подпрограммы завершения
```

5 Выводы

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

6 Список литературы

Лабораторная работа 6