

Лабораторная работа №8

Программирование цикла. Обработка аргументов командной строки

Глобин Никита Анатольевич

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
3.1	Реализация циклов в NASM	7
3.2	Обработка аргументов командной строки	8
3.3	Задание для самостоятельной работы	10
4	Выводы	12
	Список литературы	13

Список иллюстраций

3.1	photo 1	7
3.2	photo 2	7
3.3	photo 3	8
3.4	photo 4	8
3.5	photo 5	8
3.6	photo 6	9
3.7	photo 7	9
3.8	photo 8	10
3.9	photo 9	10
3.10	photo 10	11
3.11	photo 11	11

Список таблиц

1 Цель работы

Научиться работать с циклами на языке Ассемблера, а также научиться обрабатывать аргументы командной строки

2 Задание

Реализация циклов в NASM

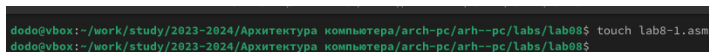
Обработка аргументов командной строки

Задание для самостоятельной работы

3 Выполнение лабораторной работы

3.1 Реализация циклов в NASM

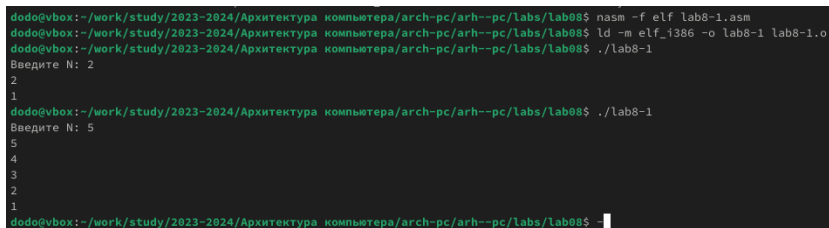
1. Для начала выполнения лабораторной работы создадим рабочую директорию и файл lab8-1.asm (рис. 3.1).



```
dodo@vbox: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/arh--pc/labs/Lab08$ touch lab8-1.asm
dodo@vbox: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/arh--pc/labs/Lab08$
```

Рис. 3.1: photo 1

2. Далее впишем программу и запустим её (рис. 3.2).



```
dodo@vbox: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/arh--pc/labs/Lab08$ nasm -f elf lab8-1.asm
dodo@vbox: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/arh--pc/labs/Lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
dodo@vbox: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/arh--pc/labs/Lab08$ ./lab8-1
Введите N: 2
2
1
dodo@vbox: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/arh--pc/labs/Lab08$ ./lab8-1
Введите N: 5
5
4
3
2
1
dodo@vbox: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/arh--pc/labs/Lab08$
```

Рис. 3.2: photo 2

3. немного перепишем программу и запустим (рис. 3.3).

```
dodo@vbox:~/work/study/2023-2024/Архитектура компьютера/arch-pc/arh--pc/labs/lab08$ nasm -f elf lab8-1.asm
dodo@vbox:~/work/study/2023-2024/Архитектура компьютера/arch-pc/arh--pc/labs/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
dodo@vbox:~/work/study/2023-2024/Архитектура компьютера/arch-pc/arh--pc/labs/lab08$ ./lab8-1
Введите N: 3
2
1
0
Ошибка сегментирования (образ памяти сброшен на диск)
dodo@vbox:~/work/study/2023-2024/Архитектура компьютера/arch-pc/arh--pc/labs/lab08$
```

Рис. 3.3: photo 3

всё работает корректно

3.2 Обработка аргументов командной строки

1. создаём новый файл и пишем в нём программу (рис. 3.4).

```
%include 'in_out.asm'
SECTION .text
global _start
_start:
    pop ecx ; Извлекаем из стека в `ecx` количество
             ; аргументов (первое значение в стеке)
    pop edx ; Извлекаем из стека в `edx` имя программы
             ; (второе значение в стеке)
    sub ecx, 1 ; Уменьшаем `ecx` на 1 (количество
             ; аргументов без названия программы)
    next:
    cmp ecx, 0 ; проверяем, есть ли еще аргументы
    jz _end ; если аргументов нет выходим из цикла
             ; (переход на метку `_end`)
    pop eax ; иначе извлекаем аргумент из стека
    call printf ; вызываем функцию печати
    loop next ; переход к обработке следующего
             ; аргумента (переход на метку `next`)
    _end:
    call quit
```

Рис. 3.4: photo 4

2. Компилируем программу и запускаем её (рис. 3.5).

```
1
dodo@vbox:~/work/study/2023-2024/Архитектура компьютера/arch-pc/arh--pc/labs/lab08$ touch lab8-2.asm
dodo@vbox:~/work/study/2023-2024/Архитектура компьютера/arch-pc/arh--pc/labs/lab08$ nasm -f elf lab8-2.asm
dodo@vbox:~/work/study/2023-2024/Архитектура компьютера/arch-pc/arh--pc/labs/lab08$ ld -m elf_i386 -o lab8-2 lab8-2.o
dodo@vbox:~/work/study/2023-2024/Архитектура компьютера/arch-pc/arh--pc/labs/lab08$ ./lab8-2 аргумент1 аргумент2 'ар
ргумент3'
2
3
dodo@vbox:~/work/study/2023-2024/Архитектура компьютера/arch-pc/arh--pc/labs/lab08$
```

Рис. 3.5: photo 5

3. создаём новый файл и пишем в нём программу (рис. 3.6).

```
%include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
SECTION .text
global _start
_start:
pop ecx ; Извлекаем из стека в `ecx` количество
; аргументов (первое значение в стеке)
pop edx ; Извлекаем из стека в `edx` имя программы
; (второе значение в стеке)
sub ecx,1 ; Уменьшаем `ecx` на 1 (количество
; аргументов без названия программы)
mov esi, 0 ; Используем `esi` для хранения
; промежуточных сумм
next:
cmp ecx,0h ; проверяем, есть ли еще аргументы
jz _end ; если аргументов нет выходим из цикла
; (переход на метку `_end`)
pop eax ; иначе извлекаем следующий аргумент из стека
call atoi ; преобразуем символ в число
add esi,eax ; добавляем к промежуточной сумме
; след. аргумент `esi=esi+eax`
loop next ; переход к обработке следующего аргумента
_end:
mov eax, msg ; вывод сообщения "Результат: "
call sprint
mov eax, esi ; записываем сумму в регистр `eax`
call iprintLF ; печать результата
call quit ; завершение программы
```

Рис. 3.6: photo 6

4. Компилируем программу и запускаем её (рис. 3.7).

```
dodo@vbox:~/work/study/2023-2024/Архитектура компьютера/arch-pc/arh--pc/labs/lab08$ touch lab8-3.asm
dodo@vbox:~/work/study/2023-2024/Архитектура компьютера/arch-pc/arh--pc/labs/lab08$ nasm -f elf lab8-3.asm
dodo@vbox:~/work/study/2023-2024/Архитектура компьютера/arch-pc/arh--pc/labs/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
dodo@vbox:~/work/study/2023-2024/Архитектура компьютера/arch-pc/arh--pc/labs/lab08$ ./main 12 13 7 10 5
bash: ./main: Нет такого файла или каталога
dodo@vbox:~/work/study/2023-2024/Архитектура компьютера/arch-pc/arh--pc/labs/lab08$ ./lab8-3 12 13 7 10 5
Результат: 47
dodo@vbox:~/work/study/2023-2024/Архитектура компьютера/arch-pc/arh--pc/labs/lab08$
```

Рис. 3.7: photo 7

5. Изменим текст программы из листинга 8.3 для вычисления произведения аргументов командной строки(рис. 3.8).

```

#include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
SECTION .text
global _start
_start:
  pop ecx ; Извлекаем из стека в `ecx` количество
           ; аргументов (первое значение в стеке)
  pop edx ; Извлекаем из стека в `edx` имя программы
           ; (второе значение в стеке)
  sub ecx,1 ; Уменьшаем `ecx` на 1 (количество
           ; аргументов без названия программы)
  mov esi, 1 ; Используем `esi` для хранения
           ; промежуточных сумм
next:
  cmp ecx,0h ; проверяем, есть ли еще аргументы
  jz _end ; если аргументов нет выходим из цикла
           ; (переход на метку `_end`)
  pop eax ; иначе извлекаем следующий аргумент из стека
  call atoi ; преобразуем символ в число
  mul esi
  mov esi,eax
  loop next ; переход к обработке следующего аргумента
_end:
  mov ebx, eax
  mov eax, msg ; вывод сообщения "Результат: "
  call sprint
  mov eax, ebx ; записываем сумму в регистр `eax`
  call iprintf ; печать результата
  call quit ; завершение программы

```

Рис. 3.8: photo 8

6. Компилируем программу и запускаем её (рис. 3.9).

```

Результат: 54600
dodo@vbox:~/work/study/2023-2024/Архитектура компьютера/arch-pc/arh--pc/labs/lab08$ nasm -f elf lab8-3.asm
dodo@vbox:~/work/study/2023-2024/Архитектура компьютера/arch-pc/arh--pc/labs/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
dodo@vbox:~/work/study/2023-2024/Архитектура компьютера/arch-pc/arh--pc/labs/lab08$ ./lab8-3 12 13 7 10 5
Результат: 54600
dodo@vbox:~/work/study/2023-2024/Архитектура компьютера/arch-pc/arh--pc/labs/lab08$ 

```

Рис. 3.9: photo 9

3.3 Задание для самостоятельной работы

1. Пишем программу для решения 9 задания (рис. 3.10).

```

%include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
msg2 db "Функция: f(x)=10x-4"
SECTION .text
global _start
_start:
pop ecx ; Извлекаем из стека в `ecx` количество
; аргументов (первое значение в стеке)
pop edx ; Извлекаем из стека в `edx` имя программы
; (второе значение в стеке)
sub ecx,1 ; Уменьшаем `ecx` на 1 (количество
; аргументов без названия программы)
mov esi, 0 ; Используем `esi` для хранения
; промежуточных сумм
next:
cmp ecx,0h ; проверяем, есть ли еще аргументы
jz _end ; если аргументов нет выходим из цикла
; (переход на метку `_end`)
pop eax ; иначе извлекаем следующий аргумент из стека
call atoi ; преобразуем символ в число
mov ebx, 10
mul ebx
sub eax, 4
add esi,eax ; добавляем к промежуточной сумме
; след. аргумент `esi=esi+eax`
loop next ; переход к обработке следующего аргумента
_end:
mov eax, msg2
call sprintf
mov eax, msg ; вывод сообщения "Результат: "
call sprintf
mov eax, esi ; записываем сумму в регистр `eax`
call iprintLF ; печать результата
call quit ; завершение программы

```

Рис. 3.10: photo 10

2. Компилируем программу и запускаем её (рис. 3.11).

```

Результат: 54600
dodo@vbox: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/arh--pc/labs/lab08$ touch task1v9.asm
dodo@vbox: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/arh--pc/labs/lab08$ nasm -f elf task1v9.asm
dodo@vbox: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/arh--pc/labs/lab08$ ld -m elf_i386 -o task1v9 task1v9.o
dodo@vbox: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/arh--pc/labs/lab08$ ./task1v9 1 2 3 4
Результат: 84

```

Рис. 3.11: photo 11

4 Выводы

В результате выполнения лабораторной работы были получены навыки работы с циклами и обработкой аргументов из командной строки. Были написаны программы, использующие все вышеописанные аспекты

Список литературы