

Отчёт по лабораторной работе 8

Программирование цикла. Обработка аргументов командной строки.

Останин Владислав Александрович НПМбв-01-21

Содержание

1 Цель работы

Целью работы является приобретение навыков написания программ с использованием циклов и обработкой аргументов командной строки..

2 Выполнение лабораторной работы

1. Создайте каталог для программ лабораторной работы № 8, перейдите в него и создайте файл lab8-1.asm
2. Введите в файл lab8-1.asm текст программы из листинга 8.1. Создайте исполняемый файл и проверьте его работу.

```
%include 'in_out.asm'
SECTION .data
msg1 db 'Введите N: ',0h
SECTION .bss
N: resb 10
SECTION .text
global _start
_start:
; ----- Вывод сообщения 'Введите N: '
mov eax,msg1
call sprint
; ----- Ввод 'N'
mov ecx, N
mov edx, 10
call sread
; ----- Преобразование 'N' из символа в число
mov eax,N
call atoi
mov [N],eax
; ----- Организация цикла
mov ecx,[N] ; Счетчик цикла, `ecx=N`
label:
mov [N],ecx
mov eax,[N]
call iprintLF ; Вывод значения `N`
loop label ; `ecx=ecx-1` и если `ecx` не `0`
; переход на `label`
call quit
```

Figure 1: Файл lab8-1.asm

```

irectory
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab08$ nasm -f elf lab8-1.asm
(failed reverse-i-search)`: ld -m elf_i386 -o lab8-1 lab8-1.o
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
(reverse-i-search)`a': ld -m elf_i386 -o lab8-1 lab8-1.o
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab08$ ls
in_out.asm  lab8-1  lab8-1.asm  lab8-1.o
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab08$ ./lab8-1
Введите N: 4
4
3
2
1
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab08$

```

Figure 2: Работа программы lab8-1.asm

3. Данный пример показывает, что использование регистра `ecx` в теле цикла `loop` может привести к некорректной работе программы. Измените текст программы добавив изменение значение регистра `ecx` в цикле: Создайте исполняемый файл и проверьте его работу. Какие значения принимает регистр `ecx` в цикле? Соответствует ли число проходов цикла значению `N`, введенному с клавиатуры?

Программа запускает бесконечный цикл при нечетном `N` и выводит только нечетные числа при четном `N`.

variant.asm	lab8-1.asm
<pre> _start: ; ----- Вывод сообщения 'Введите N: ' mov eax,msg1 call sprint ; ----- Ввод 'N' mov ecx, N mov edx, 10 call sread ; ----- Преобразование 'N' из символа в число mov eax,N call atoi mov [N],eax ; ----- Организация цикла mov ecx,[N] ; Счетчик цикла, `ecx=N` label: sub ecx,1 ; `ecx=ecx-1` mov [N],ecx mov eax,[N] call iprintLF loop label ; переход на `label` call quit </pre>	

Figure 3: Файл lab8-1.asm

```


3
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab08$ nasm -f elf
f lab8-1.asm
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab08$ ld -m elf_
i386 -o lab8-1 lab8-1.o
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab08$ ./lab8-1
Введите N: 4
3
1
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab08$

```

Figure 4: Работа программы lab8-1.asm

4. Для использования регистра `ecx` в цикле и сохранения корректности работы программы можно использовать стек. Внесите изменения в текст программы добавив команды `push` и `pop` (добавления в стек и извлечения из стека) для сохранения значения счетчика цикла `loop`. Создайте исполняемый файл и проверьте его работу. Соответствует ли в данном случае число проходов цикла значению `N` введенному с клавиатуры?

Программа выводит числа от `N-1` до `0`, число проходов цикла соответствует `N`.

Открыть ▾  lab8-1.asm
~/studies/computer_architecture/lab08

variant.asm lab8-1.asm

```
_start:
; ----- Вывод сообщения 'Введите N: '
mov eax,msg1
call sprint
; ----- Ввод 'N'
mov ecx, N
mov edx, 10
call sread
; ----- Преобразование 'N' из символа в число
mov eax,N
call atoi
mov [N],eax
; ----- Организация цикла
mov ecx,[N] ; Счетчик цикла, `ecx=N`
label:
label:
push ecx ; добавление значения ecx в стек
sub ecx,1
mov [N],ecx
mov eax,[N]
call iprintLF
pop ecx ; извлечение значения ecx из стека
loop label
call quit
```

Figure 5: Файл lab8-1.asm

```
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab08$ nasm -f elf lab8-1.asm
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab08$ ./lab8-1
Введите N: 4
3
2
1
0
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab08$
```

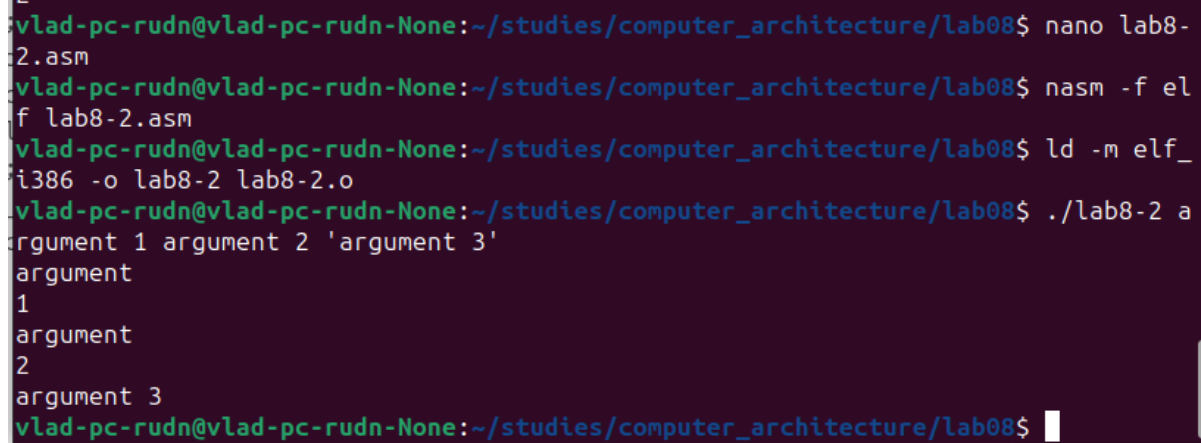
Figure 6: Работа программы *lab8-1.asm*

5. Создайте файл *lab9-2.asm* в каталоге `~/work/arch-pc/lab08` и введите в него текст программы из листинга 8.2. Создайте исполняемый файл и запустите его, указав аргументы. Сколько аргументов было обработано программой?

Программа обработала 5 аргументов.

```
%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 sprintf ; вызываем функцию печати
    loop next ; переход к обработке следующего
              ; аргумента (переход на метку `next`)
    _end:
    call quit
```

Figure 7: Файл lab8-2.asm



```
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab08$ nano lab8-2.asm
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab08$ nasm -f elf lab8-2.asm
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab08$ ld -m elf_i386 -o lab8-2 lab8-2.o
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab08$ ./lab8-2 argument 1 argument 2 'argument 3'
argument 1
argument 2
argument 3
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab08$
```

Figure 8: Работа программы lab8-2.asm

6. Рассмотрим еще один пример программы которая выводит сумму чисел, которые передаются в программу как аргументы.

```
%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
```

Figure 9: Файл lab8-3.asm

```
argument
2
argument 3
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab08$ nano lab9-
3.asm
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab08$ nasm -f elf
lab8-3.asm
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab08$ ld -m elf_
i386 -o lab8-3 lab8-3.o
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab08$ ./lab8-3 1
1 1
Результат: 3
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab08$ ./lab8-3 1
2 3
Результат: 6
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab08$
```

Figure 10: Работа программы lab8-3.asm

7. Измените текст программы из листинга 8.3 для вычисления произведения аргументов командной строки.

```
%include 'in_out.asm'
```

```
SECTION .data
```

```
msg db "Результат: ",0
```

```
SECTION .text
```

```
global _start
```

```
|
```

```
_start:
```

```
    pop ecx          ; Извлекаем из стека в `ecx` количество
аргументов
    pop edx          ; Извлекаем из стека в `edx` имя программы
    sub ecx, 1       ; Уменьшаем `ecx` на 1 (количество аргументов
без названия программы)
    mov esi, 1       ; Инициализируем `esi` как 1 для вычисления
произведения
```


```
next:
```

```
    cmp ecx, 0       ; Проверяем, есть ли еще аргументы
    jz _end          ; Если аргументов нет, выходим из цикла
    pop eax          ; Извлекаем следующий аргумент из стека
    call atoi        ; Преобразуем символ в число
    imul esi, eax     ; Умножаем промежуточное произведение на
аргумент
    dec ecx          ; Уменьшаем счетчик аргументов
    jmp next         ; Переход к обработке следующего аргумента
```

```
_end:
```

```
    mov eax, msg     ; Вывод сообщения "Результат: "
    call sprint
    mov eax, esi      ; Записываем произведение в регистр `eax`
    call iprintLF    ; Печать результата
    call quit        ; Завершение программы
```

Figure 11: Файл lab8-3.asm



```
i386 -o lab8-3 lab8-3.o
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab08$ ./lab8-3 1
2 3 4
Результат: 10
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab08$ ^C
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab08$ nano lab8-
3.asm
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab08$ nasm -f elf
lab8-3.asm
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab08$ ld -m elf_
i386 -o lab8-3 lab8-3.o
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab08$ ./lab8-3 1
2 3 4
Результат: 24
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab08$ 
call quit ; Завершение программы
```

Figure 12: Работа программы lab8-3.asm

8. Напишите программу, которая находит сумму значений функции $f(x)$ для $x = x_1, x_2, \dots, x_n$, т.е. программа должна выводить значение $f(x_1) + f(x_2) + \dots + f(x_n)$. Значения x передаются как аргументы. Вид функции $f(x)$ выбрать из таблицы 8.1 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу на нескольких наборах x .

для варианта 13 $f(x) = 12x - 7$

Открыть ▾



lab8-4.asm

~/studies/computer_architecture/lab08



variant.asm

lab8-4.asm

```
%include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
fx: db 'f(x)=12x-7 ',0

SECTION .text
global _start
_start:
    mov eax, fx
    call sprintf
    pop ecx          ; Извлекаем из стека количество аргументов
    pop edx          ; Извлекаем из стека имя программы
    sub ecx, 1       ; Уменьшаем на 1, так как не считаем имя
    программы
    mov esi, 0       ; Инициализируем `esi` для хранения
    промежуточной суммы

next:
    cmp ecx, 0
    jz _end
    pop eax          ; Извлекаем следующий аргумент из стека
    call atoi        ; Преобразуем символ в число
    imul eax, 12      ; Умножаем аргумент на 12
    sub eax, 7        ; Вычитаем 7 из результата
    add esi, eax      ; Добавляем результат к промежуточной сумме
    dec ecx          ; Уменьшаем счетчик аргументов
    jmp next         ; Переходим к следующему аргументу

_end:
    mov eax, msg      ; Вывод сообщения "Результат: "
    call sprintf
    mov eax, esi       ; Записываем сумму в `eax`
    call iprintLF      ; Печать результата
    call quit          ; Завершение программы
```

Figure 13: Файл lab8-4.asm

```
4
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab08$ nano lab8-4.asm
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab08$ nasm -f elf lab8-4.asm
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab08$ ld -m elf_i386 -o lab8-4 lab8-4.o
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab08$ ./lab8-4 1 2 3 4 5
f(x)=12x-7
Результат: 145
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab08$
```

mov eax, esi ; Записываем сумму в eax

Figure 14: Работа программы lab8-4.asm

3 Выводы

Освоили работы со стеком, циклом и аргументами на ассемблере nasm.