

Отчёт по лабораторной работе №8

Дисциплина: Архитектура компьютеров

Лазарев Даниил Михайлович

Содержание

1	Цель работы	4
2	Теоретическое введение	5
3	Выполнение лабораторной работы	6
4	Выполнение самостоятельной работы	16
5	Выводы	19

Список иллюстраций

3.1	Создание файла в каталоге	6
3.2	Код программы в файле	7
3.3	Преобразование в исполняемый файл	8
3.4	Измененный код программы	8
3.5	Преобразование измененного файла	9
3.6	Изменения исходного текста	9
3.7	Преобразование файла	10
3.8	Текст программы в файле	11
3.9	Преобразование файла в исполняемый	11
3.10	Текст программы в файле	12
3.11	Преобразование файла в исполняемый	13
3.12	Исправленный текст программы	14
3.13	Преобразование файла	15
4.1	Таблица вариантов	16
4.2	Код программы	17
4.3	Преобразование файла	18

1 Цель работы

Приобретение навыков написания программ с использованием циклов и обработкой аргументов командной строки.

2 Теоретическое введение

Стек – это структура данных, организованная по принципу LIFO («Last In – First Out»). Основной функцией стека является функция сохранения адресов возврата и передачи аргументов. Стек имеет вершину, адрес последнего добавленного элемента, который хранится в регистре. Для стека существует две основные операции:

- добавление элемента в вершину стека (push);
- извлечение элемента из вершины стека (pop).

3 Выполнение лабораторной работы

Создадим каталог для программ лаб. работы н.8, перейдем в него и создадим файл “lab8-1.asm” (рис. 3.1)



```
dmlazarev@dmlazarev:~$ mkdir ~/work/arch-pc/lab08
dmlazarev@dmlazarev:~$ cd ~/work/arch-pc/lab08
dmlazarev@dmlazarev:~/work/arch-pc/lab08$ touch lab8-1.asm
dmlazarev@dmlazarev:~/work/arch-pc/lab08$
```

Рис. 3.1: Создание файла в каталоге

Введем в созданный файл текст программы из предложенного нам листинга 8.1 (рис. 3.2)

```

%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

```

Рис. 3.2: Код программы в файле

Создадим исполняемый файл и запустим его, предварительно скопировав из предыдущей лаб. работы файл “in_out.asm” для корректной работы (рис. 3.3)

```

dmlazarev@dmlazarev:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
dmlazarev@dmlazarev:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
dmlazarev@dmlazarev:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 10
10
9
8
7
6
5
4
3
2
1

```

Рис. 3.3: Преобразование в исполняемый файл

Далее изменим часть текста. (рис. 3.5)

```

; ----- 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
call quit

```

Рис. 3.4: Измененный код программы

Преобразуем в исполняемый файл и проверим правильность выполнения. (рис. ??)


```

dmlazarev@dmlazarev:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
dmlazarev@dmlazarev:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
dmlazarev@dmlazarev:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 10
9
7
5
3
1
dmlazarev@dmlazarev:~/work/arch-pc/lab08$

```

Рис. 3.5: Преобразование измененного файла

Изменим текст листинга повторно. (рис. 3.6)

```

; ----- Организация цикла
mov ecx,[N] ; Счетчик цикла, `ecx=N`
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

```

Рис. 3.6: Изменения исходного текста

Преобразуем файл в исполняемый и проверим правильность выполнения. (рис. 3.7)

```
dmlazarev@dmlazarev:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
dmlazarev@dmlazarev:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
dmlazarev@dmlazarev:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 10
9
8
7
6
5
4
3
2
1
0
dmlazarev@dmlazarev:~/work/arch-pc/lab08$
```

Рис. 3.7: Преобразование файла

Создадим файл “lab8-2.asm” и вставим в него предложенный нам листинг 8.2 (рис. 3.8)

```

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

```

Рис. 3.8: Текст программы в файле

Преобразуем файл “lab8-2.asm” в исполняемый и проверим правильность выполнения. (рис. 3.9)

```

dmlazarev@dmlazarev:~/work/arch-pc/lab08$ nasm -f elf lab8-2.asm
dmlazarev@dmlazarev:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-2 lab8-2.o
dmlazarev@dmlazarev:~/work/arch-pc/lab08$ ./lab8-2 2 4 6
2
4
6
dmlazarev@dmlazarev:~/work/arch-pc/lab08$

```

Рис. 3.9: Преобразование файла в исполняемый

Создадим файл “lab8-3.asm” и вставим в него предложенный нам листинг 8.3.
(рис. 3.10)

```
%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. 3.10: Текст программы в файле

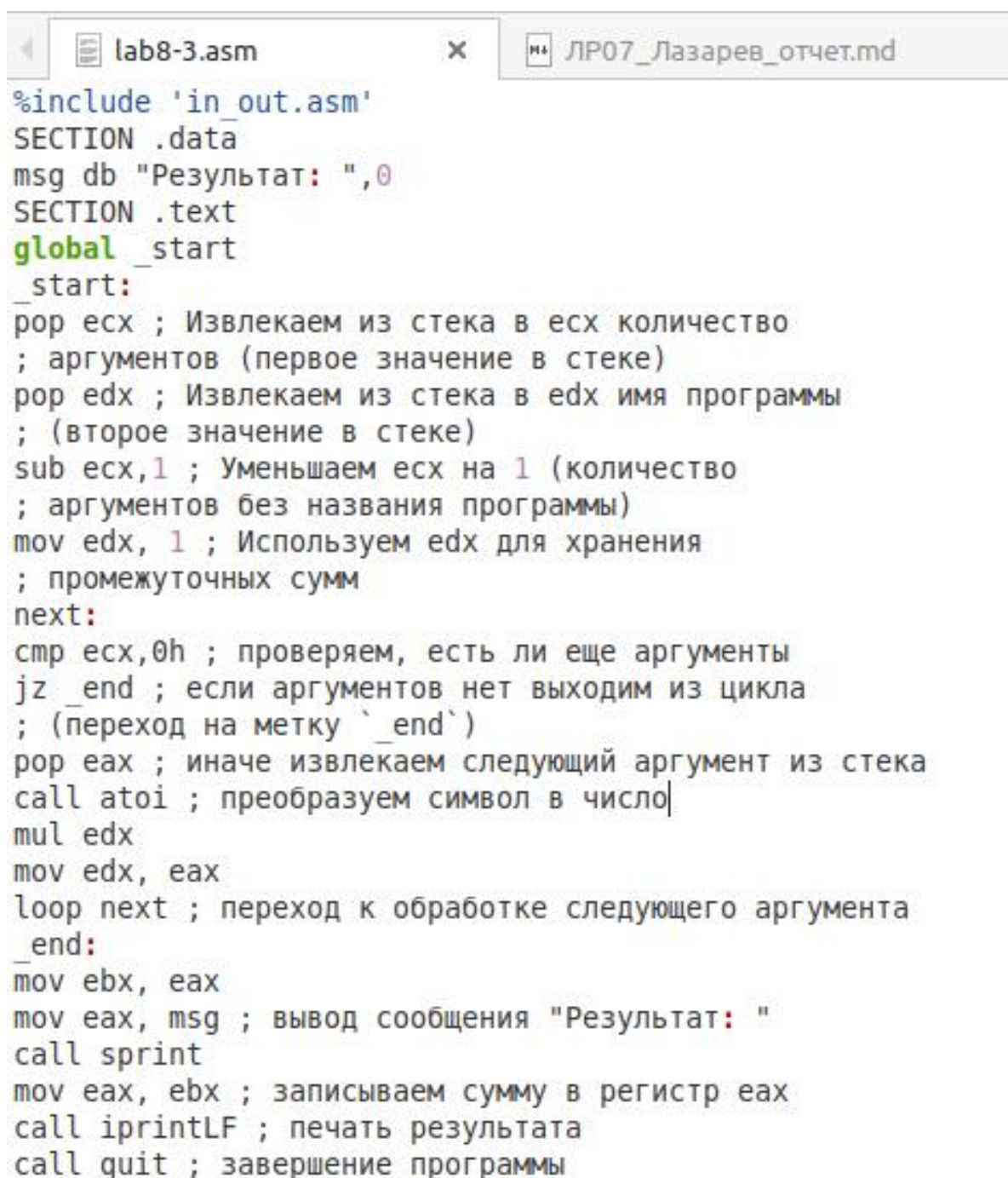
Преобразуем файл “lab8-3.asm” в исполняемый и проверим правильность

выполнения. (рис. 3.11)

```
dmlazarev@dmlazarev:~/work/arch-pc/lab08$ touch lab8-3.asm
dmlazarev@dmlazarev:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
dmlazarev@dmlazarev:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
dmlazarev@dmlazarev:~/work/arch-pc/lab08$ ./lab8-3 5 5 5 5 5
Результат: 25
dmlazarev@dmlazarev:~/work/arch-pc/lab08$
```

Рис. 3.11: Преобразование файла в исполняемый

Изменим текст листинга 8.3 так, чтобы аргументы после подставления в функцию не складывались, а умножались. (рис. 3.12)



```
lab8-3.asm x ЛР07_Лазарев_отчет.md
%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 edx, 1 ; Используем edx 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 edx
    mov edx, 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 iprintLF ; печать результата
    call quit ; завершение программы
```

Рис. 3.12: Исправленный текст программы

Преобразуем файл в исполняемый и проверим работоспособность (рис. 3.13)


```
dmlazarev@dmlazarev:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
dmlazarev@dmlazarev:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
dmlazarev@dmlazarev:~/work/arch-pc/lab08$ ./lab8-3 5 5 7
Результат: 175
dmlazarev@dmlazarev:~/work/arch-pc/lab08$
```

Рис. 3.13: Преобразование файла

4 Выполнение самостоятельной работы

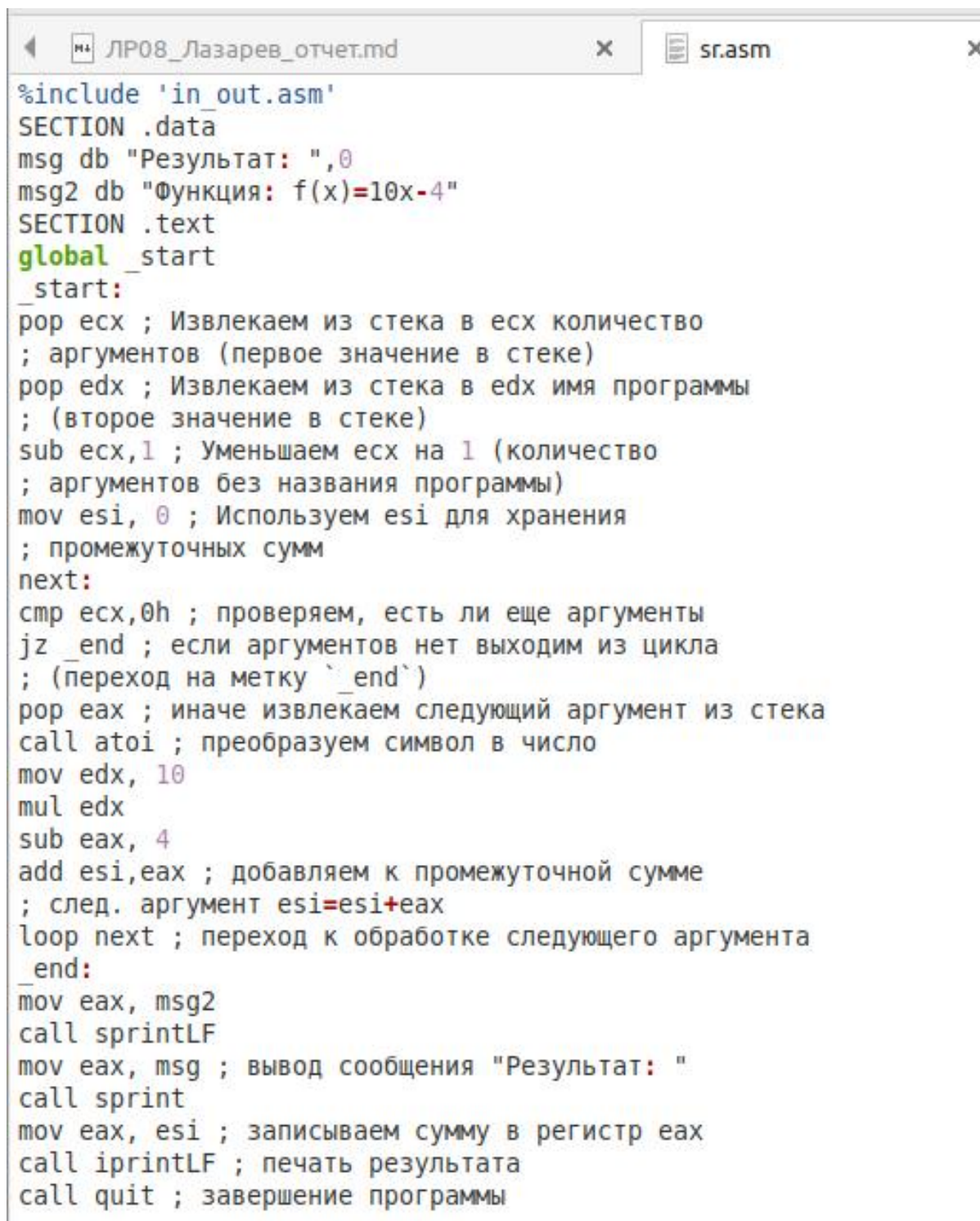
Основываясь на результате файла “variant.asm” из лаб. работы н.6 выберем из таблицы 8.1 9 номер варианта. (рис. 4.1)

Номер варианта	$f(x)$	Номер варианта	$f(x)$
1	$2x + 15$	11	$15x + 2$
2	$3x - 1$	12	$15x - 9$
3	$10x - 5$	13	$12x - 7$
4	$2(x - 1)$	14	$7(x + 1)$
5	$4x + 3$	15	$6x + 13$
6	$4x - 3$	16	$30x - 11$
7	$3(x + 2)$	17	$10(x - 1)$
8	$7 + 2x$	18	$17 + 5x$
9	$10x - 4$	19	$8x - 3$
10	$5(2 + x)$	20	$3(10 + x)$

Рис. 4.1: Таблица вариантов

Самостоятельно напишем код так, чтобы после подставления в функцию “10х-

4” все переменные суммировались. (рис. 4.2)



```
%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 edx, 10
    mul edx
    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. 4.2: Код программы

Преобразуем написанный нами файл в исполняемый и проверим правильность выполнения. (рис. 4.3)

```
dmlazarev@dmlazarev:~/work/arch-pc/lab08$ nasm -f elf sr.asm
dmlazarev@dmlazarev:~/work/arch-pc/lab08$ ld -m elf_i386 -o sr sr.o
dmlazarev@dmlazarev:~/work/arch-pc/lab08$ ./sr 0 2
Функция:  $f(x)=10x-4$ 
Результат: 12
dmlazarev@dmlazarev:~/work/arch-pc/lab08$
```

Рис. 4.3: Преобразование файла

5 Выводы

В ходе лабораторной работы мы освоили навыки написания программ с использованием циклов и обработкой аргументов командной строки.