

Отчёт по лабораторной работе №9

Дисциплина: Архитектура компьютера

Мишина Анастасия Алексеевна

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выполнение заданий самостоятельной работы	17
4	Выводы	20

Список иллюстраций

2.1	Создание каталога и файла	6
2.2	Работа с файлом lab9-1.asm	8
2.3	Работа с измененным файлом lab9-1.asm	10
2.4	Работа с измененным файлом lab9-1.asm	12
2.5	Работа с файлом lab9-2.asm	13
2.6	Работа с файлом lab9-3.asm	15
2.7	Работа с измененным файлом lab9-3.asm	16
3.1	Тестирование программы lab9my.asm	19

Список таблиц

1 Цель работы

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

2 Выполнение лабораторной работы

Для начала создадим каталог для программ 9-ой лабораторной работы, перейдем в него и создадим файл lab9-1.asm (рис. 2.1).

```
[aamishina@fedora ~]$ mkdir ~/work/arch-pc/lab09  
[aamishina@fedora ~]$ cd ~/work/arch-pc/lab09  
[aamishina@fedora lab09]$ touch lab9-1.asm  
[aamishina@fedora lab09]$
```

Рис. 2.1: Создание каталога и файла

Вводим текст программы из листинга 9.1 в наш файл. Создадим и запустим исполняемый файл, удостоверимся в его работе (рис. 2.2).

Программа lab9-1.asm:

```
-----  
; Программа вывода значений регистра 'ecx'  
-----  
  
%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

```

```

[aamishina@fedora lab09]$ nasm -f elf lab9-1.asm
[aamishina@fedora lab09]$ ld -m elf_i386 -o lab9-1 lab9-1.o
[aamishina@fedora lab09]$ ./lab9-1
Введите N: 8
8
7
6
5
4
3
2
1
[aamishina@fedora lab09]$

```

Рис. 2.2: Работа с файлом lab9-1.asm

Меняем текст программы. Создаем исполняемый файл, видим, что при вводе четного числа выводятся только нечетные. При вводе нечетного числа видим бесконечный вывод чисел (рис. 2.3).

Измененная программа lab9-1.asm:

```

;-----
; Программа вывода значений регистра 'ecx'
;-----
%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:
sub ecx,1 ; `ecx=ecx-1`
mov [N],ecx
mov eax,[N]
call iprintLF
loop label
; переход на `label`
call quit

```

```

[aamishina@fedora lab09]$ nasm -f elf lab9-1.asm
[aamishina@fedora lab09]$ ld -m elf_i386 -o lab9-1 lab9-1.o
[aamishina@fedora lab09]$ ./lab9-1
Введите N: 8
7
5
3
1
[aamishina@fedora lab09]$ ./lab9-1
Введите N: 7
6
4
2
0
4294967294
4294967292
4294967290
4294967288
4294967286
4294967284

```

Рис. 2.3: Работа с измененным файлом lab9-1.asm

Теперь будем использовать стек для сохранения значения счетчика в цикле и корректности работы программы (рис. 2.4). Как видно, при вводе значения N выводятся числа от N-1 до 0. Число проходов цикла соответствует значению N.

Измененная программа lab9-1.asm:

```

;-----
; Программа вывода значений регистра 'ecx'
;-----

%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:
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
; переход на `label`
call quit

```

```

[aamishina@fedora lab09]$ nasm -f elf lab9-1.asm
[aamishina@fedora lab09]$ ld -m elf_i386 -o lab9-1 lab9-1.o
[aamishina@fedora lab09]$ ./lab9-1
Введите N: 8
7
6
5
4
3
2
1
0
[aamishina@fedora lab09]$ ./lab9-1
Введите N: 7
6
5
4
3
2
1
0
[aamishina@fedora lab09]$

```

Рис. 2.4: Работа с измененным файлом lab9-1.asm

Создаем файл lab9-2.asm, вставляем в него текст программы из листинга 9.2 для обработки аргументов командной строки. Создаем и запускаем исполняемый файл, проверяем его работу (рис. 2.5). Программа обработала 5 аргументов.

Программа lab9-2.asm:

```

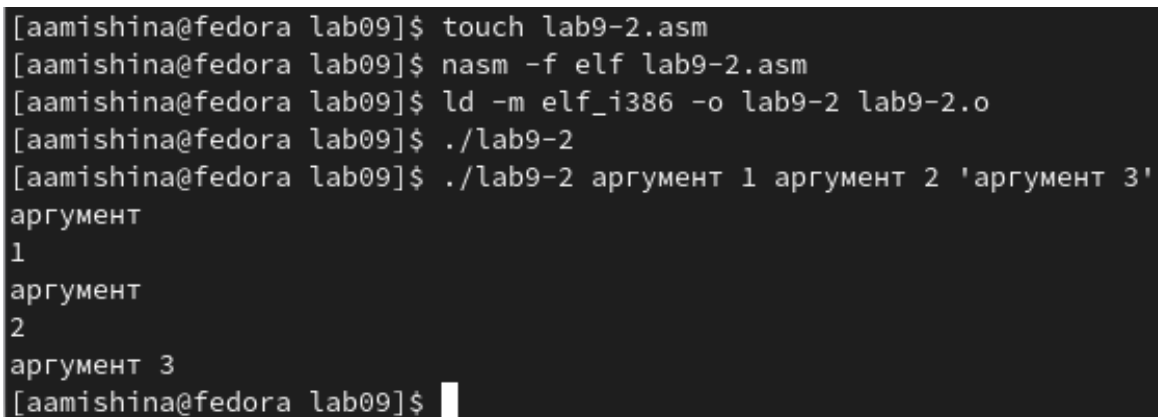
;-----
; Обработка аргументов командной строки
;-----
%include 'in_out.asm'
SECTION .text
global _start
_start:

```

```

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

```



```

[aamishina@fedora lab09]$ touch lab9-2.asm
[aamishina@fedora lab09]$ nasm -f elf lab9-2.asm
[aamishina@fedora lab09]$ ld -m elf_i386 -o lab9-2 lab9-2.o
[aamishina@fedora lab09]$ ./lab9-2
[aamishina@fedora lab09]$ ./lab9-2 аргумент 1 аргумент 2 'аргумент 3'
аргумент
1
аргумент
2
аргумент 3
[aamishina@fedora lab09]$

```

Рис. 2.5: Работа с файлом lab9-2.asm

Далее создадим файл lab9-3.asm и введем в него текст программы вычисления суммы аргументов командной строки из листинга 9.3. (рис. 2.6). Удостоверяемся в работе программы.

Программа lab9-3.asm:

```
%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

```
[aamishina@fedora lab09]$ touch lab9-3.asm
[aamishina@fedora lab09]$ nasm -f elf lab9-3.asm
[aamishina@fedora lab09]$ ld -m elf_i386 -o lab9-3 lab9-3.o
[aamishina@fedora lab09]$ ./lab9-3
Результат: 0
[aamishina@fedora lab09]$ ./lab9-3 1 2 3 9 1
Результат: 16
[aamishina@fedora lab09]$
```

Рис. 2.6: Работа с файлом lab9-3.asm

Изменим текст файла lab9-3.asm, чтобы теперь программа считала произведение введенных аргументов (рис. 2.7).

Измененная программа lab9-3.asm:

```
%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
mov ebx, eax
mov eax, esi
mul ebx
mov 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

```

```

[aamishina@fedora lab09]$ nasm -f elf lab9-3.asm
[aamishina@fedora lab09]$ ld -m elf_i386 -o lab9-3 lab9-3.o
[aamishina@fedora lab09]$ ./lab9-3 10 3 2 1
Результат: 60
[aamishina@fedora lab09]$ ./lab9-3
Результат: 1
[aamishina@fedora lab09]$ █

```

Рис. 2.7: Работа с измененным файлом lab9-3.asm

3 Выполнение заданий самостоятельной работы

Напишем программу, которая находит сумму значений функций для введенных аргументов. Согласно моему варианту (13) мне следовало использовать функцию $f(x) = 12x - 7$. Тестируем работу программы. Программа отработала успешно (рис. 3.1).

Программа lab9my.asm:

```
%include 'in_out.asm'

SECTION .data
msg db "Результат: ", 0
func: db 'f(x) = 12x - 7', 0

SECTION .text
global _start
_start:
mov eax, func
call sprintf

pop ecx
pop edx
sub ecx, 1
```

```
mov esi, 0

next:
cmp ecx, 0h
jz _end

pop eax
call atoi

mov ebx, 12
mul ebx

sub eax, 7
add esi, eax

loop next

_end:
mov eax, msg
call sprint

mov eax, esi
call iprintLF

call quit
```

```
[aamishina@fedora lab09]$ nasm -f elf lab9my.asm
[aamishina@fedora lab09]$ ld -m elf_i386 -o lab9my lab9my.o
[aamishina@fedora lab09]$ ./lab9my 1
f(x) = 12x - 7
Результат: 5
[aamishina@fedora lab09]$ ./lab9my 4 2 5
f(x) = 12x - 7
Результат: 111
[aamishina@fedora lab09]$ ./lab9my 22 41 8 12
f(x) = 12x - 7
Результат: 968
[aamishina@fedora lab09]$
```

Рис. 3.1: Тестирование программы lab9my.asm

4 Выводы

В ходе выполнения данной лабораторной работы я приобрела навыки написания программ с использованием циклов и обработкой аргументов командной строки. Вся моя работа была записана и прокомментирована мной в данной лабораторной.