

Отчёт по лабораторной работе №8

Дисциплина: архитектура компьютера

Мусатова Екатерина Викторовна

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Самостоятельная работа	15
4	Выводы	18

Список иллюстраций

2.1	Создание необходимого каталога и файла	6
2.2	Ввод текста программы	7
2.3	Проверка	7
2.4	Изменение программы	8
2.5	Проверка	8
2.6	Изменение программы	9
2.7	Проверка	10
2.8	Создание файла	10
2.9	Ввод текста	11
2.10	Проверка	11
2.11	Создание файла	12
2.12	Ввод текста	12
2.13	Проверка	13
2.14	Изменение программы	13
2.15	Проверка	14
3.1	Написание программы	16
3.2	Проверка	17

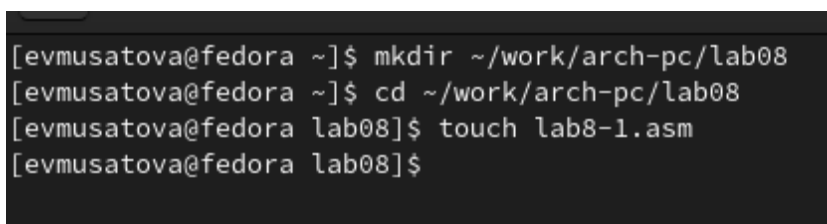
Список таблиц

1 Цель работы

Получение навыков по организации циклов и работе со стеком на языке NASM.

2 Выполнение лабораторной работы

Создаю каталог для программ лабораторной работы № 8, перехожу в него и создаю файл lab8-1.asm (рис. 2.1).

A terminal window with a dark background and light-colored text. It shows a series of four commands being executed in a shell. The first command creates a directory, the second changes the current directory to that directory, the third creates a new file, and the fourth shows the prompt after the file is created.

```
[evmusatova@fedora ~]$ mkdir ~/work/arch-pc/lab08
[evmusatova@fedora ~]$ cd ~/work/arch-pc/lab08
[evmusatova@fedora lab08]$ touch lab8-1.asm
[evmusatova@fedora lab08]$
```

Рис. 2.1: Создание необходимого каталога и файла

Ввожу в созданный файл текст программы из листинга 8.1 (рис. 2.2).

```

1 ;-----
2 ; Программа вывода значений регистра 'ecx'
3 ;-----
4 %include 'in_out.asm'
5 SECTION .data
6 msg1 db 'Введите N: ',0h
7 SECTION .bss
8 N: resb 10
9 SECTION .text
10 global _start
11 _start:
12 ; ----- Вывод сообщения 'Введите N: '
13 mov eax,msg1
14 call printf
15 ; ----- Ввод 'N'
16 mov ecx, N
17 mov edx, 10
18 call scanf
19 ; ----- Преобразование 'N' из символа в число
20 mov eax,N
21 call atoi
22 mov [N],eax
23 ; ----- Организация цикла
24 mov ecx,[N] ; Счетчик цикла, `ecx=N`
25 label:
26 mov [N],ecx
27 mov eax,[N]
28 call printf ; Вывод значения `N`
29 loop label ; `ecx=ecx-1` и если `ecx` не '0'
30 ; переход на `label`
31 call quit

```

Рис. 2.2: Ввод текста программы

Создаю исполняемый файл и проверяю его работу (рис. 2.3). Вижу что цикл выводит цифры от 3 до 1.

```

[evmusatova@fedora lab08]$ ./lab8-1
Введите N: 3
3
2
1
[evmusatova@fedora lab08]$

```

Рис. 2.3: Проверка

Изменяю текст программы добавляя изменение значения регистра ecx в

цикле (рис. 2.4).

```
25 label:
26 mov [N],ecx
27 mov eax,[N]
28 call iprintLF ; Вывод значения `N`
29 loop label ; `ecx=ecx-1` и если `ecx` не '0'
30 ; переход на `label`
31 sub ecx,1 ; `ecx=ecx-1`
32 mov [N],ecx
33 mov eax,[N]
34 call iprintLF
35 loop label
36 call quit
```

Рис. 2.4: Изменение программы

Создаю исполняемый файл и проверяю его работу (рис. 2.5). Число проходов цикла не соответствует значению N введенному с клавиатуры.

```
4294503540
4294503539
4294503538
4294503537
4294503536
4294503535
4294503534
4294503533
4294503532
4294503531
4294503530
4294503529
4294503528
4294^C
[evmusatova@fedora lab08]$
```

Рис. 2.5: Проверка

Вношу изменения в текст программы добавив команды push и pop (рис. 2.6).


```

1  ;-----
2  ; Программа вывода значений регистра 'ecx'
3  ;-----
4  %include 'in_out.asm'
5  SECTION .data
6  msg1 db 'Введите N: ',0h
7  SECTION .bss
8  N: resb 10
9  SECTION .text
10 global _start
11 _start:
12 ; ---- Вывод сообщения 'Введите N: '
13 mov eax,msg1
14 call sprint
15 ; ---- Ввод 'N'
16 mov ecx, N
17 mov edx, 10
18 call sread
19 ; ---- Преобразование 'N' из символа в число
20 mov eax,N
21 call atoi
22 mov [N],eax
23 ; ----- Организация цикла
24 mov ecx,[N] ; Счетчик цикла, `ecx=N`
25 label:
26 push ecx ; добавление значения ecx в стек
27 sub ecx,1 ; `ecx=ecx-1`
28 mov [N],ecx
29 mov eax,[N]
30 call iprintf
31 pop ecx ; извлечение значения ecx из стека
32 loop label
33 call quit

```

Рис. 2.6: Изменение программы

Создаю исполняемый файл и проверяю его работу (рис. 2.7). В данном случае число подходов цикла соответствует значению N введенному с клавиатуры.

```
[evmusatova@fedora lab08]$ nasm -f elf lab8-1.asm
[evmusatova@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[evmusatova@fedora lab08]$ ./lab8-1
Введите N: 3
2
1
0
[evmusatova@fedora lab08]$
```

Рис. 2.7: Проверка

Создаю файл lab8-2.asm (рис. 2.8).

```
[evmusatova@fedora lab08]$ touch lab8-2.asm
[evmusatova@fedora lab08]$
```

Рис. 2.8: Создание файла

Ввожу в файл текст программы из листинга 8.2 (рис. 2.9).

```

1  %include 'in_out.asm'
2  SECTION .text
3  global _start
4  _start:
5  pop ecx ; Извлекаем из стека в `ecx` количество
6  ; аргументов (первое значение в стеке)
7  pop edx ; Извлекаем из стека в `edx` имя программы
8  ; (второе значение в стеке)
9  sub ecx, 1 ; Уменьшаем `ecx` на 1 (количество
10 ; аргументов без названия программы)
11 next:
12 cmp ecx, 0 ; проверяем, есть ли еще аргументы
13 jz _end ; если аргументов нет выходим из цикла
14 ; (переход на метку `_end`)
15 pop eax ; иначе извлекаем аргумент из стека
16 call printf ; вызываем функцию печати
17 loop next ; переход к обработке следующего
18 ; аргумента (переход на метку `next`)
19 _end:
20 call quit

```

Рис. 2.9: Ввод текста

Создаю исполняемый файл и проверяю его работу, указав аргументы (рис. 2.10).

```

[evmusatova@fedora lab08]$ ./lab8-2
[evmusatova@fedora lab08]$ ./lab8-2 аргумент1 аргумент 2 'аргумент 3'
аргумент1
аргумент
2
аргумент 3
[evmusatova@fedora lab08]$

```

Рис. 2.10: Проверка

Создаю файл lab8-3.asm (рис. 2.11).

```
[evmusatova@fedora lab08]$ touch lab8-3.asm  
[evmusatova@fedora lab08]$
```

Рис. 2.11: Создание файла

Ввожу в файл текст программы из листинга 8.3 (рис. 2.12).

```
1 %include 'in_out.asm'  
2 SECTION .data  
3 msg db "Результат: ",0  
4 SECTION .text  
5 global _start  
6 _start:  
7 pop ecx ; Извлекаем из стека в `ecx` количество  
8 ; аргументов (первое значение в стеке)  
9 pop edx ; Извлекаем из стека в `edx` имя программы  
10 ; (второе значение в стеке)  
11 sub ecx,1 ; Уменьшаем `ecx` на 1 (количество  
12 ; аргументов без названия программы)  
13 mov esi, 0 ; Используем `esi` для хранения  
14 ; промежуточных сумм  
15 next:  
16 cmp ecx,0h ; проверяем, есть ли еще аргументы  
17 jz _end ; если аргументов нет выходим из цикла  
18 ; (переход на метку `_end`)  
19 pop eax ; иначе извлекаем следующий аргумент из стека  
20 call atoi ; преобразуем символ в число  
21 add esi,eax ; добавляем к промежуточной сумме  
22 ; след. аргумент `esi=esi+eax`  
23 loop next ; переход к обработке следующего аргумента  
24 _end:  
25 mov eax, msg ; вывод сообщения "Результат: "  
26 call sprint  
27 mov eax, esi ; записываем сумму в регистр `eax`  
28 call iprintLF ; печать результата  
29 call quit ; завершение программы
```

Рис. 2.12: Ввод текста

Создаю исполняемый файл и проверяю его работу (рис. 2.13). Программа вычисляет сумму всех введенных аргументов.

```
[evmusatova@fedora lab08]$ nasm -f elf lab8-3.asm
[evmusatova@fedora lab08]$ ld -m elf_i386 -o lab8-3 lab8-3.o
[evmusatova@fedora lab08]$ ./lab8-3 30 12 6 8 22
Результат: 78
[evmusatova@fedora lab08]$
```

Рис. 2.13: Проверка

Изменяю текст программы для вычисления произведения аргументов командной строки (рис. 2.14).

```
1  %include "in_out.asm"
2  SECTION .data
3  msg db 'результат: '
4  SECTION .text
5  GLOBAL _start
6  _start:
7  pop ecx
8  pop edx
9  sub ecx,1
10 mov esi,1
11 next:
12 cmp ecx,0
13 jz _end
14 pop eax
15 call atoi
16 mul esi
17 mov esi, eax
18 loop next
19 _end:
20 mov eax, msg
21 call sprint
22 mov eax, esi
23 call iprintLF
24 call quit
```

Рис. 2.14: Изменение программы

Создаю исполняемый файл и проверяю работу программы (рис. 2.15).

```
[evmusatova@fedora lab08]$ nasm -f elf lab8-3.asm
[evmusatova@fedora lab08]$ ld -m elf_i386 -o lab8-3 lab8-3.o
[evmusatova@fedora lab08]$ ./lab8-3 5 3 20
результат: 300
[evmusatova@fedora lab08]$
```

Рис. 2.15: Проверка

3 Самостоятельная работа

Пишу программу, которая находит сумму значений функции (рис. 3.1).



```
lab8-3.asm
1  %include 'in_out.asm'
2
3  SECTION .data
4  f_x db "функция: 7 + 2x",0h
5  msg db 10,13,'результат: ',0h
6
7  SECTION .text
8  global _start
9
10 _start:
11 pop ecx
12 mov esi, 0
13
14 next:
15 cmp ecx,0h
16 jz _end
17 pop eax
18 call atoi
19 mov ebx, 2
20 mul ebx
21 add eax, 7
22 add esi, eax
23
24 loop next
25
26 _end:
27 mov eax, f_x
28 call sprint
29 mov eax, msg
30 call sprint
31 mov eax, esi
32 call iprintLF
33
34 call quit
```

Рис. 3.1: Написание программы

Проверяю правильность написания программы с разными аргументами (рис.

3.2).

```
результат: 500
[evmusatova@fedora lab08]$ touch lab8-4.asm
[evmusatova@fedora lab08]$ nasm -f elf lab8-4.asm
[evmusatova@fedora lab08]$ ld -m elf_i386 -o lab8-4 lab8-4.o
[evmusatova@fedora lab08]$ ./lab8-4 1 2 3 4
функция: 7 + 2x
результат: 55
[evmusatova@fedora lab08]$ ./lab8-4 5 4 7 8
функция: 7 + 2x
результат: 83
```

Рис. 3.2: Проверка

4 Выводы

Были получены навыки по организации циклов и работе со стеком на языке NASM.