

Отчёт по лабораторной работе №8

Дисциплина: Архитектура компьютера

Толстых Александра Андреевна

Содержание

1	Цель работы	4
2	Задание	5
3	Выполнение лабораторной работы	6
3.1	Реализация циклов в NASM	6
3.2	Обработка аргументов командной строки	9
3.3	Задание для самостоятельной работы	11
4	Выводы	13

Список иллюстраций

3.1	Создание каталога и файла	6
3.2	Написание текста программы	6
3.3	Запуск программы	7
3.4	Изменение программы	7
3.5	Запуск программы	8
3.6	Изменение программы	8
3.7	Запуск программы	9
3.8	Создание файла	9
3.9	Написание программы	9
3.10	Запуск программы	10
3.11	Создание файла	10
3.12	Написание программы	10
3.13	Запуск программы	10
3.14	Изменение программы	11
3.15	Запуск программы	11
3.16	Создание файла	11
3.17	Написание программы	12
3.18	Запуск программы	12

1 Цель работы

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

2 Задание

1. Реализация циклов в NASM
2. Обработка аргументов командной строки
3. Задание для самостоятельной работы

3 Выполнение лабораторной работы

3.1 Реализация циклов в NASM

Создаю каталог для программ лабораторной работы № 8, перехожу в него и создаю файл lab8-1.asm (рис. 3.1).

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

Рис. 3.1: Создание каталога и файла

Ввожу в файл lab8-1.asm текст программы из листинга 8.1 (рис. 3.2).



```
lab8-1.asm
~/work/arch-pc/lab08
Ln 35, Col 20

1 ;-----
2 ; Программа вывода значений регистра 'ecx'
3 ;-----
4
5 %include 'in_out.asm'
6
7 SECTION .data
8     msg1 db 'Введите N: ',0h
9
10 SECTION .bss
11     N: resb 10
12
13 SECTION .text
14     global _start
15 _start:
16
17 ; ----- Вывод сообщения 'Введите N: '
18     mov eax,msg1
19     call sprint
20
21 ; ----- Ввод 'N'
22     mov ecx, N
23     mov edx, 10
24     call sread
25
26 ; ----- Преобразование 'N' из символа в число
27     mov eax,N
```

Рис. 3.2: Написание текста программы

Создаю исполняемый файл и проверяю его работу. Анализируя результат, замечая, что использование регистра `ecx` в теле цикла `loop` может привести к некорректной работе программы (рис. 3.3).

```
aatolstihkh@aatolstihkh:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
aatolstihkh@aatolstihkh:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
aatolstihkh@aatolstihkh:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 5
5
4
3
2
1
aatolstihkh@aatolstihkh:~/work/arch-pc/lab08$
```

Рис. 3.3: Запуск программы

Изменяю текст программы добавив изменение значения регистра `ecx` в цикле (рис. 3.4).



```
Open  report.md  • lab8-1.asm  Ln 38, Col 22
~/work/arch-pc/lab08  • lab8-1.asm  x

23     mov ecx, 10
24     call sread
25
26 ; ---- Преобразование 'N' из символа в число
27     mov eax, N
28     call atoi
29     mov [N], eax
30
31 ; ----- Организация цикла
32     mov ecx, [N]          ; Счетчик цикла, 'ecx=N'
33
34 label:
35     sub ecx, 1
36     mov [N], ecx
37     mov eax, [N]
38     call iprintLF
39
40     loop label
41
42     call quit
```

Рис. 3.4: Изменение программы

Создаю исполняемый файл и проверяю его работу. Регистр `ecx` в цикле принимает значения, отличающиеся на 2, а также после 0 работает некорректно (рис. 3.5).

```
aatolstikhkh@aatolstikhkh:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 5
4
2
0
4294967294
4294967292
4294967290
4294967288
4294967286
4294967284
4294967282
4294967280
4294967278
4294967276
4294967274
4294967272
4294967270
```

Рис. 3.5: Запуск программы

Вношу изменения в текст программы добавив команды push и pop (добавления в стек и извлечения из стека) для сохранения значения счетчика цикла (рис. 3.6).

```
Open  lab8-1.asm  Ln 44, Col 18
~/work/arch-pc/lab08

22     mov ecx, N
23     mov edx, 10
24     call sread
25
26 ; ---- Преобразование 'N' из символа в число
27     mov eax, N
28     call atoi
29     mov [N], eax
30
31 ; ----- Организация цикла
32     mov ecx, [N]                ; Счетчик цикла, 'ecx=N'
33
34 label:
35     push ecx                    ; добавление значения ecx в стек
36     sub ecx, 1
37     mov [N], ecx
38     mov eax, [N]
39     call iprintf
40     pop ecx                     ; извлечение значения ecx из стека
41
42     loop label
43
44     call quit
```

Рис. 3.6: Изменение программы

Создаю исполняемый файл и проверяю его работу. Теперь программа работает корректно (рис. 3.7).


```

aato1stihkh@aato1stihkh:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
aato1stihkh@aato1stihkh:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
aato1stihkh@aato1stihkh:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 5
4
3
2
1
0
aato1stihkh@aato1stihkh:~/work/arch-pc/lab08$

```

Рис. 3.7: Запуск программы

3.2 Обработка аргументов командной строки

Создаю файл lab8-2.asm в каталоге ~/work/arch-pc/lab08 (рис. 3.8).

```

aato1stihkh@aato1stihkh:~/work/arch-pc/lab08$ touch lab8-2.asm
aato1stihkh@aato1stihkh:~/work/arch-pc/lab08$

```

Рис. 3.8: Создание файла

Ввожу в него текст программы из листинга 8.2 (рис. 3.9).

```

Open ▾  lab8-2.asm  Ln 26, Col 9
~/work/arch-pc/lab08

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

```

Рис. 3.9: Написание программы

Создаю исполняемый файл и запускаю его, указывая данные аргументы. Программа обрабатывает 4 аргумента - “аргумент1”, “аргумент”, “2”, “аргумент 3” (рис. 3.10).

```

aamolstihkh@aamolstihkh:~/work/arch-pc/lab08$ nasm -f elf lab8-2.asm
aamolstihkh@aamolstihkh:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-2 lab8-2.o
aamolstihkh@aamolstihkh:~/work/arch-pc/lab08$ ./lab8-2 аргумент1 аргумент 2 'аргумент 3'

аргумент1
аргумент
2
аргумент 3
aamolstihkh@aamolstihkh:~/work/arch-pc/lab08$

```

Рис. 3.10: Запуск программы

Создаю файл lab8-3.asm в каталоге ~/work/arch-pc/lab08 (рис. 3.11).

```

aamolstihkh@aamolstihkh:~/work/arch-pc/lab08$ touch lab8-3.asm
aamolstihkh@aamolstihkh:~/work/arch-pc/lab08$

```

Рис. 3.11: Создание файла

Ввожу в него текст программы из листинга 8.3 (рис. 3.12).

```

Open ▾  lab8-3.asm  Ln 32, Col 41
~/work/arch-pc/lab08

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

```

Рис. 3.12: Написание программы

Создаю исполняемый файл и запускаю его, указывая некоторые числа как аргументы. Программа их обрабатывает и выводит их сумму (рис. 3.13).

```

aamolstihkh@aamolstihkh:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
aamolstihkh@aamolstihkh:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
aamolstihkh@aamolstihkh:~/work/arch-pc/lab08$ ./lab8-3 12 13 7 10 5
Результат: 47
aamolstihkh@aamolstihkh:~/work/arch-pc/lab08$

```

Рис. 3.13: Запуск программы

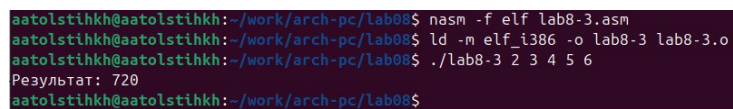
Изменяю текст программы из листинга 8.3 для вычисления произведения аргументов командной строки (рис. 3.14).



```
10     pop ecx             ; Извлекаем из стека в `ecx` количество
11                               ; аргументов (первое значение в стеке)
12     pop edx             ; Извлекаем из стека в `edx` имя программы
13                               ; (второе значение в стеке)
14     sub ecx,1           ; Уменьшаем `ecx` на 1 (количество
15                               ; аргументов без названия программы)
16     mov esi, 1
17     mov eax, 1
18
19 next:
20     cmp ecx,0h          ; проверяем, есть ли еще аргументы
21     jz _end             ; если аргументов нет выходим из цикла
22                               ; (переход на метку `_end`)
23     pop eax              ; иначе извлекаем следующий аргумент из стека
24     call atoi            ; преобразуем символ в число
25     mov ebx,eax
26     mov eax,esi
27     mul ebx
28     mov esi,eax
29     loop next           ; переход к обработке следующего аргумента
30 _end:
31     mov eax, msg         ; вывод сообщения "Результат: "
32     call sprintf
33     mov eax, esi         ; записываем сумму в регистр `eax`
34     call iprintLF        ; печать результата
35     call quit            ; завершение программы
```

Рис. 3.14: Изменение программы

Создаю исполняемый файл и запускаю его, указывая некоторые числа как аргументы. Программа их обрабатывает и выводит их произведение (рис. 3.15).



```
aatolstihkh@aatolstihkh:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
aatolstihkh@aatolstihkh:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
aatolstihkh@aatolstihkh:~/work/arch-pc/lab08$ ./lab8-3 2 3 4 5 6
Результат: 720
aatolstihkh@aatolstihkh:~/work/arch-pc/lab08$
```

Рис. 3.15: Запуск программы

3.3 Задание для самостоятельной работы

Создаю файл lab8-4.asm в каталоге ~/work/arch-pc/lab08 (рис. 3.16).



```
aatolstihkh@aatolstihkh:~/work/arch-pc/lab08$ touch lab8-4.asm
aatolstihkh@aatolstihkh:~/work/arch-pc/lab08$
```

Рис. 3.16: Создание файла

Создаю программу, которая находит сумму значений функции $f(x)=30x-11$ (16 вариант) для некоторых точек, заданных как аргументы (рис. 3.17).



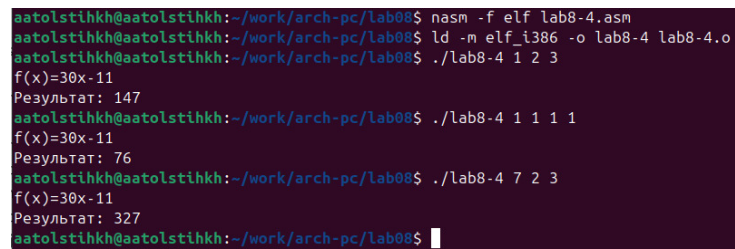
```

1 %include 'in_out.asm'
2
3 SECTION .data
4 prim DB 'f(x)=30x-11',0
5 otv DB 'Результат: ',0
6 SECTION .text
7 GLOBAL _start
8 _start:
9
10 pop ecx
11
12 pop edx
13
14 sub ecx,1
15
16 mov esi,0
17
18 mov eax,prim
19 call sprintf
20 next:
21 cmp ecx,0
22 jz _end
23
24 mov ebx,30
25 pop eax
26 call atoi
27 imul ebx

```

Рис. 3.17: Написание программы

Создаю исполняемый файл и проверяю его работу на нескольких наборах чисел (рис. 3.18).



```

aamolstihkh@aamolstihkh:~/work/arch-pc/lab08$ nasm -f elf lab8-4.asm
aamolstihkh@aamolstihkh:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-4 lab8-4.o
aamolstihkh@aamolstihkh:~/work/arch-pc/lab08$ ./lab8-4 1 2 3
f(x)=30x-11
Результат: 147
aamolstihkh@aamolstihkh:~/work/arch-pc/lab08$ ./lab8-4 1 1 1 1
f(x)=30x-11
Результат: 76
aamolstihkh@aamolstihkh:~/work/arch-pc/lab08$ ./lab8-4 7 2 3
f(x)=30x-11
Результат: 327
aamolstihkh@aamolstihkh:~/work/arch-pc/lab08$

```

Рис. 3.18: Запуск программы

4 Выводы

В результате выполнения лабораторной работы я приобрела навыки написания программ с использованием циклов и обработкой аргументов командной строки.