

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ № 8

дисциплина: *Архитектура компьютера*

Студент: Терентьев Максим Сергеевич

Группа: НКАбд-05-25

МОСКВА

2025 г.

Содержание

1.	Цель работы.....	3
2.	Порядок выполнения лабораторной работы	4
2.1.	Реализация циклов в NASM.....	4
2.2.	Обработка аргументов командной строки	6
3.	Задание для самостоятельной работы.....	8
	Выводы.....	10

1. Цель работы

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

2. Порядок выполнения лабораторной работы

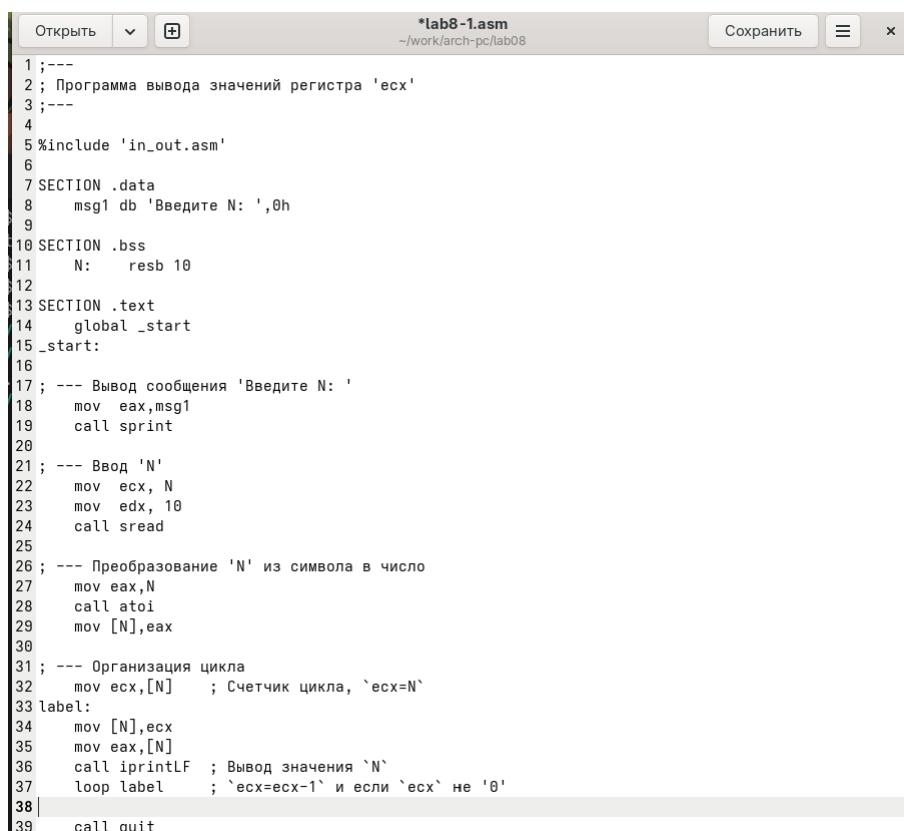
2.1. Реализация циклов в NASM

Создадим каталог для выполнения лабораторной работы, перейдем в него и сделаем файл lab8-1.asm, а также скопируем в него файл in_out.asm из прошлой лабораторной работы:

```
msterentjev@rudn:~$ mkdir ~/work/arch-pc/lab08
msterentjev@rudn:~$ cp ~/work/arch-pc/lab07/in_out.asm ~/work/arch-pc/lab08/in_out.asm
msterentjev@rudn:~$ cd ~/work/arch-pc/lab08/
msterentjev@rudn:~/work/arch-pc/lab08$ touch lab8-1.asm
msterentjev@rudn:~/work/arch-pc/lab08$ ls
in_out.asm  lab8-1.asm
msterentjev@rudn:~/work/arch-pc/lab08$
```

Рис. 1. Создание lab8-1.asm и каталога lab08

Введём в созданный файл код из листинша 8.1:



```
Открыть  *lab8-1.asm
Сохранить
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
28     call atoi
29     mov [N],eax
30
31 ; --- Организация цикла
32     mov ecx,[N]      ; Счетчик цикла, `ecx=N`
33 label:
34     mov [N],ecx
35     mov eax,[N]
36     call iprintLF   ; Вывод значения `N`
37     loop label       ; `ecx=ecx-1` и если `ecx` не `0`
38
39     call quit
```

Рис. 2. Текст программы lab8-1.asm

Теперь создадим исполняемый файл и проверим его работу:

```
msterentjev@rudn:~/work/arch-pc/lab08$ gedit lab8-1.asm
msterentjev@rudn:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
nasm: fatal: unrecognised output format `--elf' - use -hf for a list
Type nasm -h for help.
msterentjev@rudn:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
msterentjev@rudn:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
msterentjev@rudn:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 12
12
11
10
9
8
7
6
5
4
3
2
1
msterentjev@rudn:~/work/arch-pc/lab08$
```

Рис. 3. Создание исполн. файла и работа программы lab8-1.asm

Данный пример показывает, что использование регистра ecx в теле цикла loop может привести к некорректной работе программы. Изменим текст программы добавив изменение значение регистра ecx в цикле:

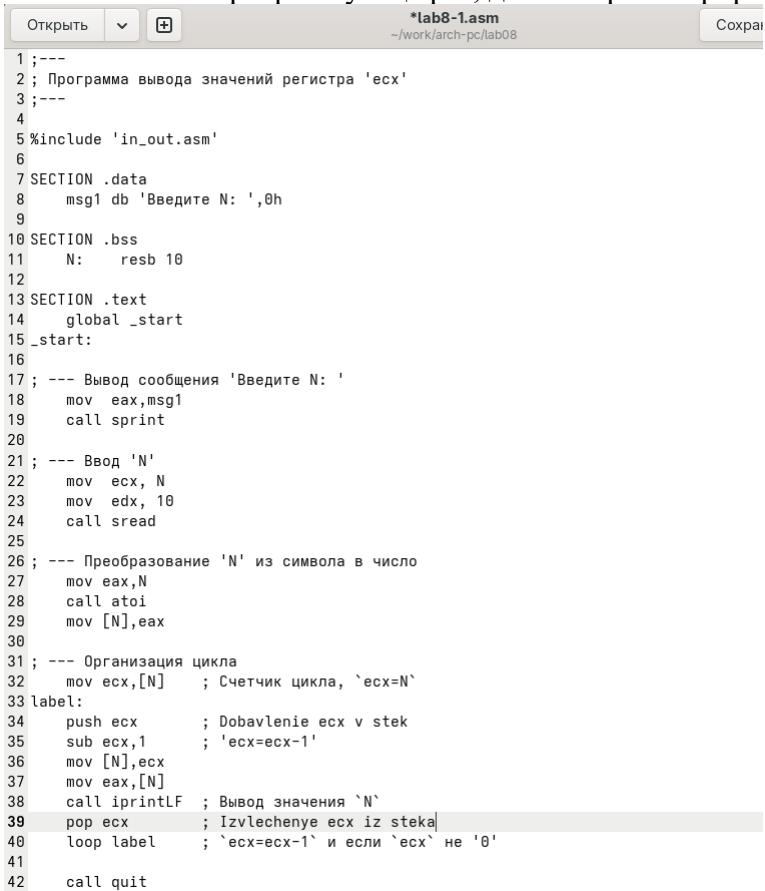
```
1 ;--  
2 ; Программа вывода значений регистра 'ecx'  
3 ;--  
4  
5 %include 'in_out.asm'  
6  
7 SECTION .data  
8     msg1 db 'Введите N: ',0h  
9  
10 SECTION .bss  
11    N:    resb 10  
12  
13 SECTION .text  
14    global _start  
15 _start:  
16  
17 ; --- Вывод сообщения 'Введите N: '  
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  
28     call atoi  
29     mov [N],eax  
30  
31 ; --- Организация цикла  
32     mov ecx,[N] ; Счетчик цикла, `ecx=N'  
33 label:  
34     sub ecx,1 ; 'ecx=ecx-1'|  
35     mov [N],ecx  
36     mov eax,[N]  
37     call iprintLF ; Вывод значения `N'  
38     loop label ; `ecx=ecx-1` и если `ecx` не '0'  
39  
40     call quit
```

Рис. 4. Измененный текст программы lab8-1.asm

Создадим исполняемый файл и проверим его работу:

```
msterentjev@rudn:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm  
msterentjev@rudn:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o  
msterentjev@rudn:~/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  
4294967268  
4294967266  
4294967264  
4294967262  
4294967260  
4294967258  
4294967256  
4294967254  
4294967252  
4294967250  
4294967248  
4294967246  
4294967244  
4294967242  
4294967240  
4294967238  
4294967236  
4294967234  
4294967232  
4294967230  
4294967228  
4294967226  
4294967224  
4294967222  
4294967220  
4294967218  
4294967216  
4294967214  
4294967212  
4294967210  
4294967208  
4294967206  
4294967204  
4294967202  
4294967200  
4294967198  
4294967196  
4294967194  
4294967192  
4294967190  
4294967188  
4294967186  
4294967184  
4294967182  
4294967180  
4294967178  
4294967176  
4294967174  
4294967172  
4294967170  
4294967168  
4294967166  
4294967164  
4294967162  
4294967160  
4294967158  
4294967156  
4294967154  
4294967152  
4294967150  
4294967148  
4294967146  
4294967144  
4294967142  
4294967140  
4294967138  
4294967136  
4294967134  
4294967132  
4294967130  
4294967128  
4294967126  
4294967124  
4294967122  
4294967120  
4294967118  
4294967116  
4294967114  
4294967112  
4294967110  
4294967108  
4294967106  
4294967104  
4294967102  
4294967100  
4294967098  
4294967096  
4294967094  
4294967092  
4294967090  
4294967088  
4294967086  
4294967084  
4294967082  
4294967080  
4294967078  
4294967076  
4294967074  
4294967072  
4294967070  
4294967068  
4294967066  
4294967064  
4294967062  
4294967060  
4294967058  
4294967056  
4294967054  
4294967052  
4294967050  
4294967048  
4294967046  
4294967044  
4294967042  
4294967040  
4294967038  
4294967036  
4294967034  
4294967032  
4294967030  
4294967028  
4294967026  
4294967024  
4294967022  
4294967020  
4294967018  
4294967016  
4294967014  
4294967012  
4294967010  
4294967008  
4294967006  
4294967004  
4294967002  
4294967000  
4294966998  
4294966996  
4294966994  
4294966992  
4294966990  
4294966988  
4294966986  
4294966984  
4294966982  
4294966980  
4294966978  
4294966976  
4294966974  
4294966972  
4294966970  
4294966968  
4294966966  
4294966964  
4294966962  
4294966960  
4294966958  
4294966956  
4294966954  
4294966952  
4294966950  
4294966948  
4294966946  
4294966944  
4294966942  
4294966940  
4294966938  
4294966936  
4294966934  
4294966932  
4294966930  
4294966928  
4294966926  
4294966924  
4294966922  
4294966920  
4294966918  
4294966916  
4294966914  
4294966912  
4294966910  
4294966908  
4294966906  
4294966904  
4294966902  
4294966900  
4294966898  
4294966896  
4294966894  
4294966892  
4294966890  
4294966888  
4294966886  
4294966884  
4294966882  
4294966880  
4294966878  
4294966876  
4294966874  
4294966872  
4294966870  
4294966868  
4294966866  
4294966864  
4294966862  
4294966860  
4294966858  
4294966856  
4294966854  
4294966852  
4294966850  
4294966848  
4294966846  
4294966844  
4294966842  
4294966840  
4294966838  
4294966836  
4294966834  
4294966832  
4294966830  
4294966828  
4294966826  
4294966824  
4294966822  
4294966820  
4294966818  
4294966816  
4294966814  
4294966812  
4294966810  
4294966808  
4294966806  
4294966804  
4294966802  
4294966800  
4294966798  
4294966796  
4294966794  
4294966792  
4294966790  
4294966788  
4294966786  
4294966784  
4294966782  
4294966780  
4294966778  
4294966776  
4294966774  
4294966772  
4294966770  
4294966768  
4294966766  
4294966764  
4294966762  
4294966760  
4294966758  
4294966756  
4294966754  
4294966752  
4294966750  
4294966748  
4294966746  
4294966744  
4294966742  
4294966740  
4294966738  
4294966736  
4294966734  
4294966732  
4294966730  
4294966728  
4294966726  
4294966724  
4294966722  
4294966720  
4294966718  
4294966716  
4294966714  
4294966712  
4294966710  
4294966708  
4294966706  
4294966704  
4294966702  
4294966700  
4294966698  
4294966696  
4294966694  
4294966692  
4294966690  
4294966688  
4294966686  
4294966684  
4294966682  
4294966680  
4294966678  
4294966676  
4294966674  
4294966672  
4294966670  
4294966668  
4294966666  
4294966664  
4294966662  
4294966660  
4294966658  
4294966656  
4294966654  
4294966652  
4294966650  
4294966648  
4294966646  
4294966644  
4294966642  
4294966640  
4294966638  
4294966636  
4294966634  
4294966632  
4294966630  
4294966628  
4294966626  
4294966624  
4294966622  
4294966620  
4294966618  
4294966616  
4294966614  
4294966612  
4294966610  
4294966608  
4294966606  
4294966604  
4294966602  
4294966600  
4294966598  
4294966596  
4294966594  
4294966592  
4294966590  
4294966588  
4294966586  
4294966584  
4294966582  
4294966580  
4294966578  
4294966576  
4294966574  
4294966572  
4294966570  
4294966568  
4294966566  
4294966564  
4294966562  
4294966560  
4294966558  
4294966556  
4294966554  
4294966552  
4294966550  
4294966548  
4294966546  
4294966544  
4294966542  
4294966540  
4294966538  
4294966536  
4294966534  
4294966532  
4294966530  
4294966528  
4294966526  
4294966524  
4294966522  
4294966520  
4294966518  
4294966516  
4294966514  
4294966512  
4294966510  
4294966508  
4294966506  
4294966504  
4294966502  
4294966500  
4294966498  
4294966496  
4294966494  
4294966492  
4294966490  
4294966488  
4294966486  
4294966484  
4294966482  
4294966480  
4294966478  
4294966476  
4294966474  
4294966472  
4294966470  
4294966468  
4294966466  
4294966464  
4294966462  
4294966460  
4294966458  
4294966456  
4294966454  
4294966452  
4294966450  
4294966448  
4294966446  
4294966444  
4294966442  
4294966440  
4294966438  
4294966436  
4294966434  
4294966432  
4294966430  
4294966428  
4294966426  
4294966424  
4294966422  
4294966420  
4294966418  
4294966416  
4294966414  
4294966412  
4294966410  
4294966408  
4294966406  
4294966404  
4294966402  
4294966400  
4294966398  
4294966396  
4294966394  
4294966392  
4294966390  
4294966388  
4294966386  
4294966384  
4294966382  
4294966380  
4294966378  
4294966376  
4294966374  
4294966372  
4294966370  
4294966368  
4294966366  
4294966364  
4294966362  
4294966360  
4294966358  
4294966356  
4294966354  
4294966352  
4294966350  
4294966348  
4294966346  
4294966344  
4294966342  
4294966340  
4294966338  
4294966336  
4294966334  
4294966332  
4294966330  
4294966328  
4294966326  
4294966324  
4294966322  
4294966320  
4294966318  
4294966316  
4294966314  
4294966312  
4294966310  
4294966308  
4294966306  
4294966304  
4294966302  
4294966300  
4294966298  
4294966296  
4294966294  
4294966292  
4294966290  
4294966288  
4294966286  
4294966284  
4294966282  
4294966280  
4294966278  
4294966276  
4294966274  
4294966272  
4294966270  
4294966268  
4294966266  
4294966264  
4294966262  
4294966260  
4294966258  
4294966256  
4294966254  
4294966252  
4294966250  
4294966248  
4294966246  
4294966244  
4294966242  
4294966240  
4294966238  
4294966236  
4294966234  
4294966232  
4294966230  
4294966228  
4294966226  
4294966224  
4294966222  
4294966220  
4294966218  
4294966216  
4294966214  
4294966212  
4294966210  
4294966208  
4294966206  
4294966204  
4294966202  
4294966200  
4294966198  
4294966196  
4294966194  
4294966192  
4294966190  
4294966188  
4294966186  
4294966184  
4294966182  
4294966180  
4294966178  
4294966176  
4294966174  
4294966172  
4294966170  
4294966168  
4294966166  
4294966164  
4294966162  
4294966160  
4294966158  
4294966156  
4294966154  
4294966152  
4294966150  
4294966148  
4294966146  
4294966144  
4294966142  
4294966140  
4294966138  
4294966136  
4294966134  
4294966132  
4294966130  
4294966128  
4294966126  
4294966124  
4294966122  
4294966120  
4294966118  
4294966116  
4294966114  
4294966112  
4294966110  
4294966108  
4294966106  
4294966104  
4294966102  
4294966100  
4294966098  
4294966096  
4294966094  
4294966092  
4294966090  
4294966088  
4294966086  
4294966084  
4294966082  
4294966080  
4294966078  
4294966076  
4294966074  
4294966072  
4294966070  
4294966068  
4294966066  
4294966064  
4294966062  
4294966060  
4294966058  
4294966056  
4294966054  
4294966052  
4294966050  
4294966048  
4294966046  
4294966044  
4294966042  
4294966040  
4294966038  
4294966036  
4294966034  
4294966032  
4294966030  
4294966028  
4294966026  
4294966024  
4294966022  
4294966020  
4294966018  
4294966016  
4294966014  
4294966012  
4294966010  
4294966008  
4294966006  
4294966004  
4294966002  
4294966000  
4294965998  
4294965996  
4294965994  
4294965992  
4294965990  
4294965988  
4294965986  
4294965984  
4294965982  
4294965980  
4294965978  
4294965976  
4294965974  
4294965972  
4294965970  
4294965968  
4294965966  
4294965964  
4294965962  
4294965960  
4294965958  
4294965956  
4294965954  
4294965952  
4294965950  
4294965948  
4294965946  
4294965944  
4294965942  
4294965940  
4294965938  
4294965936  
4294965934  
4294965932  
4294965930  
4294965928  
4294965926  
4294965924  
4294965922  
4294965920  
4294965918  
4294965916  
4294965914  
4294965912  
4294965910  
4294965908  
4294965906  
4294965904  
4294965902  
4294965900  
4294965898  
4294965896  
4294965894  
4294965892  
4294965890  
4294965888  
4294965886  
4294965884  
4294965882  
4294965880  
4294965878  
4294965876  
4294965874  
4294965872  
4294965870  
4294965868  
4294965866  
4294965864  
4294965862  
4294965860  
4294965858  
4294965856  
4294965854  
4294965852  
4294965850  
4294965848  
4294965846  
4294965844  
4294965842  
4294965840  
4294965838  
4294965836  
4294965834  
4294965832  
4294965830  
4294965828  
4294965826  
4294965824  
4294965822  
4294965820  
4294965818  
4294965816  
4294965814  
4294965812  
4294965810  
4294965808  
4294965806  
4294965804  
4294965802  
4294965800  
4294965798  
4294965796  
4294965794  
4294965792  
4294965790  
4294965788  
4294965786  
4294965784  
4294965782  
4294965780  
4294965778  
4294965776  
4294965774  
4294965772  
4294965770  
4294965768  
4294965766  
4294965764  
4294965762  
4294965760  
4294965758  
4294965756  
4294965754  
4294965752  
4294965750  
4294965748  
4294965746  
4294965744  
4294965742  
4294965740  
4294965738  
4294965736  
4294965734  
4294965732  
4294965730  
4294965728  
4294965726  
4294965724  
4294965722  
4294965720  
4294965718  
4294965716  
4294965714  
4294965712  
4294965710  
4294965708  
4294965706  
4294965704  
4294965702  
4294965700  
4294965698  
4294965696  
4294965694  
4294965692  
4294965690  
4294965688  
4294965686  
4294965684  
4294965682  
4294965680  
4294965678  
4294965676  
4294965674  
4294965672  
4294965670  
4294965668  
4294965666  
4294965664  
4294965662  
4294965660  
4294965658  
4294965656  
4294965654  
4294965652  
4294965650  
4294965648  
4294965646  
4294965644  
4294965642  
4294965640  
4294965638  
4294965636  
4294965634  
4294965632  
4294965630  
4294965628  
4294965626  
4294965624  
4294965622  
4294965620  
4294965618  
4294965616  
4294965614  
4294965612  
4294965610  
4294965608  
4294965606  
4294965604  
4294965602  
4294965600  
4294965598  
4294965596  
4294965594  
4294965592  
4294965590  
4294965588  
4294965586  
4294965584  
4294965582  
4294965580  
4294965578  
4294965576  
4294965574  
4294965572  
4294965570  
4294965568  
429496
```

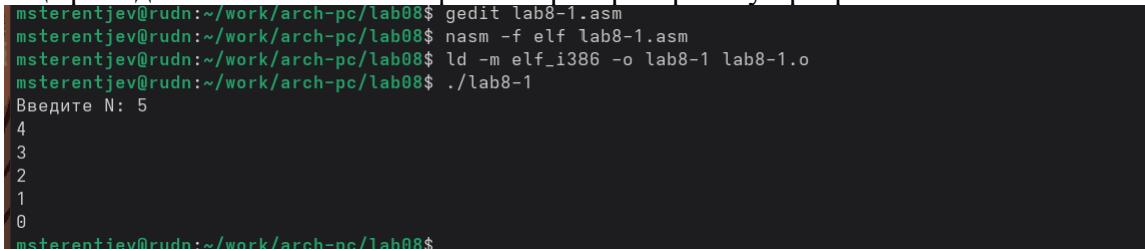
Изменим программу еще раз, добавив push и pop:



```
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  
28     call atoi  
29     mov [N],eax  
30  
31 ; --- Организация цикла  
32     mov ecx,[N]      ; Счетчик цикла, `ecx=N`  
33 label:  
34     push ecx          ; Добавление ecx в стек  
35     sub ecx,1         ; `ecx=ecx-1`  
36     mov [N],ecx  
37     mov eax,[N]  
38     call iprintLF   ; Вывод значения `N`  
39     pop ecx           ; Извлечение ecx из стека  
40     loop label        ; `ecx=ecx-1` и если `ecx` не `0`  
41  
42     call quit
```

Рис. 6. Текст программы lab8-1.asm с использованием стека

Еще раз сделаем исполняемый файл и проверим работу программы:



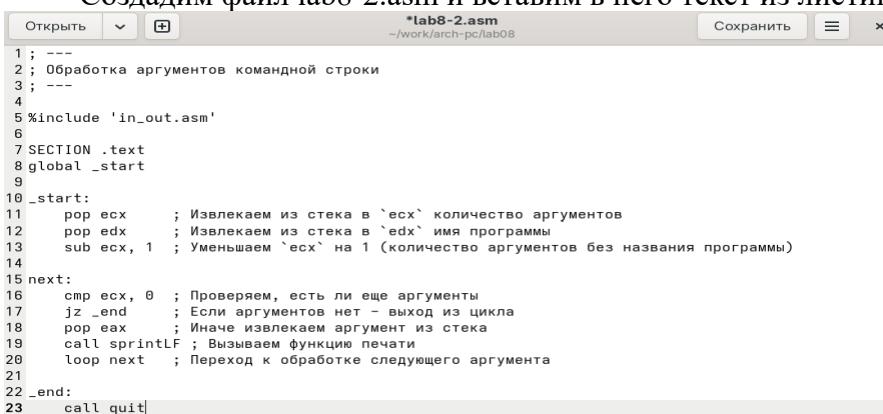
```
msterentjev@rudn:~/work/arch-pc/lab08$ gedit lab8-1.asm  
msterentjev@rudn:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm  
msterentjev@rudn:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o  
msterentjev@rudn:~/work/arch-pc/lab08$ ./lab8-1  
Введите N: 5  
5  
4  
3  
2  
1  
0
```

Рис. 7. Работа lab8-1.asm с использованием стека

Как видим, программа снова работает корректно. Значение N введенное с клавиатуры совпадает с количеством проходов цикла.

2.2. Обработка аргументов командной строки

Создадим файл lab8-2.asm и вставим в него текст из листинга 8.2:



```
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 аргументов  
12     pop edx          ; Извлекаем из стека в `edx` имя программы  
13     sub ecx, 1         ; Уменьшаем `ecx` на 1 (количество аргументов без названия программы)  
14  
15 next:  
16     cmp ecx, 0         ; Проверяем, есть ли еще аргументы  
17     jz _end            ; Если аргументов нет - выход из цикла  
18     pop eax            ; Иначе извлекаем аргумент из стека  
19     call sprintLF     ; Вызываем функцию печати  
20     loop next          ; Переход к обработке следующего аргумента  
21  
22 _end:  
23     call quit
```

Рис. 8. Текст программы lab8-2.asm из листинга 8.2

Создадим исполняемый файл и запустим его, указав аргументы:

```
msterentjev@rudn:~/work/arch-pc/lab08$ nasm -f elf lab8-2.asm
msterentjev@rudn:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-2 lab8-2.o
msterentjev@rudn:~/work/arch-pc/lab08$ ./lab8-2
msterentjev@rudn:~/work/arch-pc/lab08$ ./lab8-2 аргумент1 аргумент2 'аргумент3'
аргумент1
аргумент
2
аргумент 3
msterentjev@rudn:~/work/arch-pc/lab08$
```

Рис. 9. Работа lab8-2.asm с аргументами

Как видим, программа обработала 4 аргумента.

Рассмотрим еще один пример программы которая выводит сумму чисел, которые передаются в программу как аргументы. Создадим файл lab8-3.asm и введём в него текст программы из листинга 8.3:

```
1 %include 'in_out.asm'
2
3 SECTION .data
4     msg db "Результат: ",0
5
6 SECTION .text
7 global _start
8
9 _start:
10    pop ecx      ; Извлекаем из стека в `ecx` количество аргументов
11    pop edx      ; Извлекаем из стека в `edx` имя программы
12    sub ecx,1    ; Уменьшаем `ecx` на 1 (количество аргументов без названия программы)
13    mov esi, 0    ; Используем `esi` для хранения промежуточных сумм
14
15 next:
16    cmp ecx,0h    ; Проверяем, есть ли еще аргументы
17    jz _end        ; Если аргументов нет – выходим из цикла
18    pop eax        ; Иначе извлекаем следующий аргумент из стека
19    call atoi      ; Преобразуем символ в число
20    add esi,eax   ; Добавляем к промежуточной сумме следующий аргумент `esi=esi+eax`
21    loop next      ; Переход к обработке следующего аргумента
22
23 _end:
24    mov eax, msg   ; Вывод сообщения "Результат: "
25    call sprint
26    mov eax, esi   ; Записываем сумму в регистр `eax`
27    call iprintfLF ; Печать результата
28    call quit      ; Завершение программы
```

Рис. 10. Текст программы lab8-3.asm

Создадим исполняемый файл и проверим его работу:

```
msterentjev@rudn:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
msterentjev@rudn:~/work/arch-pc/lab08$ ./lab8-3 12 8 10 5
Результат: 35
msterentjev@rudn:~/work/arch-pc/lab08$
```

Рис. 11. Работа программы lab8-3.asm

Как видим, программа работает корректно: $12+8+10+5$ действительно равно 35.

Изменим программу для работы с умножением:

```
1 %include 'in_out.asm'
2
3 SECTION .data
4     msg db "Результат: ",0
5
6 SECTION .text
7 global _start
8
9 _start:
10    pop ecx      ; Извлекаем из стека в `ecx` количество аргументов
11    pop edx      ; Извлекаем из стека в `edx` имя программы
12    sub ecx,1    ; Уменьшаем `ecx` на 1 (количество аргументов без названия программы)
13    mov esi, 1    ; Используем `esi` для хранения произведения (начальное значение 1)
14
15 next:
16    cmp ecx,0h    ; Проверяем, есть ли еще аргументы
17    jz _end        ; Если аргументов нет – выходим из цикла
18    pop eax        ; Иначе извлекаем следующий аргумент из стека
19    call atoi      ; Преобразуем символ в число
20    imul esi, eax  ; Умножаем промежуточное произведение на следующий аргумент
21    loop next      ; Переход к обработке следующего аргумента
22
23 _end:
24    mov eax, msg   ; Вывод сообщения "Результат: "
25    call sprint
26    mov eax, esi   ; Записываем произведение в регистр `eax`
27    call iprintfLF ; Печать результата
28    call quit      ; Завершение программы
```

Рис. 12. Измененный код lab8-3.asm под умножение

Создадим исполняемый файл и проверим его работу:

```
msterentjev@rudn:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
msterentjev@rudn:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
msterentjev@rudn:~/work/arch-pc/lab08$ ./lab8-3 8 5 10
Результат: 400
msterentjev@rudn:~/work/arch-pc/lab08$
```

Рис. 13. Работа lab8-3.asm с умножением

Как видим, все работает корректно.

3. Задание для самостоятельной работы

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_i передаются как аргументы. Вид функции $f(x)$ выбрать из таблицы 8.1 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 6. Создайте исполняемый файл и проверьте его работу на нескольких наборах $x = x_1, x_2, \dots, x_n$.

Мой вариант – 11, значит функция – $15x + 2$.

Создадим файл labobus8.asm и вставим в него код получившейся программы:

```
Открыть  *labobus8.asm
~/work/arch-pc/lab08
Сохранить  ×

1 %include 'in_out.asm'
2
3 SECTION .data
4 func_msg db "Функция: f(x)=15x+2",0
5 result_msg db "Результат: ",0
6
7 SECTION .text
8 global _start
9
10 _start:
11 pop ecx          ; Извлекаем количество аргументов
12 pop edx          ; Извлекаем имя программы
13 sub ecx, 1       ; Уменьшаем количество аргументов (без имени программы)
14 mov esi, 0        ; Инициализируем сумму результатов функции
15
16 ; Выводим информацию о функции
17 mov eax, func_msg
18 call sprintLF
19
20 cmp ecx, 0        ; Проверяем, есть ли аргументы
21 jz _end            ; Если нет аргументов – выходим
22
23 next:
24 pop eax          ; Извлекаем аргумент из стека
25 call atoi          ; Преобразуем в число
26
27 ; Вычисляем f(x) = 15x + 2
28 mov ebx, eax      ; Сохраняем x в ebx
29 imul eax, 15       ; Умножаем на 15: eax = 15x
30 add eax, 2         ; Прибавляем 2: eax = 15x + 2
31
32 add esi, eax      ; Добавляем результат к общей сумме
33
34 loop next          ; Переход к следующему аргументу
35
36 _end:
37 mov eax, result_msg ; Вывод сообщения "Результат: "
38 call sprint
39 mov eax, esi        ; Записываем сумму в eax
40 call iprintLF       ; Выводим результат
41 call quit           ; Завершаем программу
```

Рис. 14. Код программы labobus8.asm

Создадим исполняемый файл и проверим работу программы:

```
msterentjev@rudn:~/work/arch-pc/lab08$ nasm -f elf labobus8.asm
msterentjev@rudn:~/work/arch-pc/lab08$ ld -m elf_i386 -o labobus8 labobus8.o
msterentjev@rudn:~/work/arch-pc/lab08$ ./labobus8 1 2 3 4
Функция: f(x)=15x+2
Результат: 158
msterentjev@rudn:~/work/arch-pc/lab08$ ./labobus8 0 4 1 5
Функция: f(x)=15x+2
Результат: 158
msterentjev@rudn:~/work/arch-pc/lab08$ ./labobus8 1 1 1 1
Функция: f(x)=15x+2
Результат: 68
msterentjev@rudn:~/work/arch-pc/lab08$
```

Рис. 15. Работа программы labobus8.asm

Первый случай: $15*1+2 + 15*2+2 + 15*3+2 + 15*4+2 = 17 + 32 + 47 + 62 = 158 \Rightarrow$ верно

Второй случай: $15*0+2 + 15*4+2 + 15*1+2 + 15*5+2 = 2 + 62 + 17 + 77 = 158 \Rightarrow$ верно

Третий случай: $15*1+2 + 15*1+2 + 15*1+2 + 15*1+2 = 17 + 17 + 17 + 17 = 68 \Rightarrow$ верно

Как мы видим, программа работает корректно.

Выводы

Приобретены навыки работы с циклами и аргументами командной строки в ассемблере.
Реализована программа вычисления суммы значений функции для заданных аргументов.