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: Силантьева Анастасия

Группа: НКАбд-05-25

№ ст. билета: 1032253541

МОСКВА

2025 г.

Содержание

Список иллюстраций

1. Цель работы

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

2. Выполнение лабораторной работы

2.1 Реализация циклов в NASM

Создаю каталог для программ лабораторной работы №8. (Рис. 1)

Рис. 1 Создание каталога

Копирую в созданный файл программу из листинга. (Рис. 2)

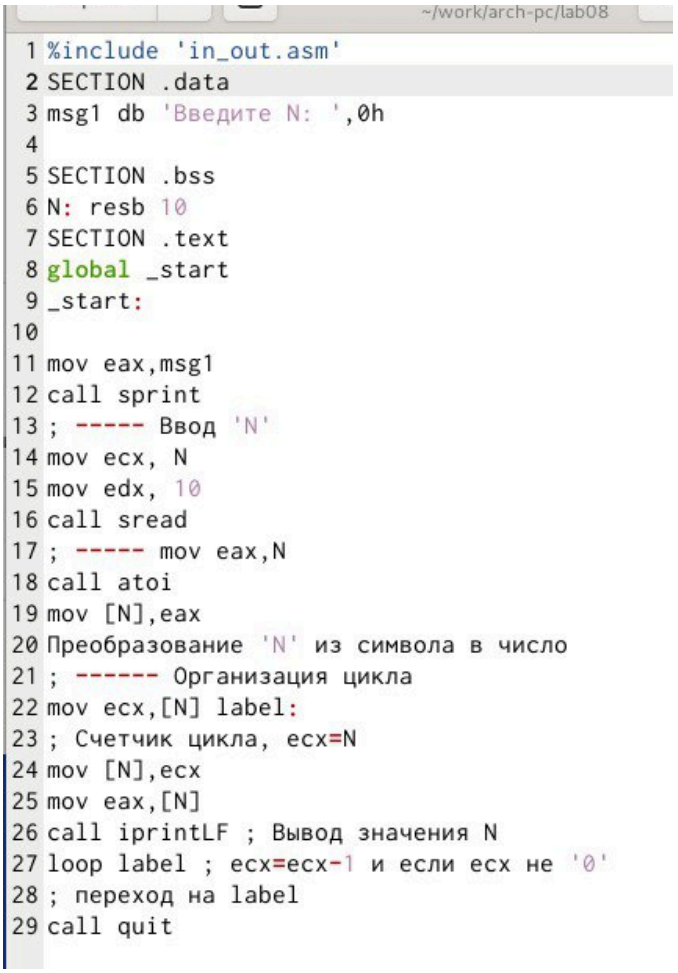

Рис. 2 Копирование программы из листинга

Запускаю программу, она показывает работу циклов в NASM. (рис. 3).

Рис. 3 Запуск программы

Заменяю программу изначальную так, что в теле цикла я изменяю значение регистра ecx. (рис. 4).

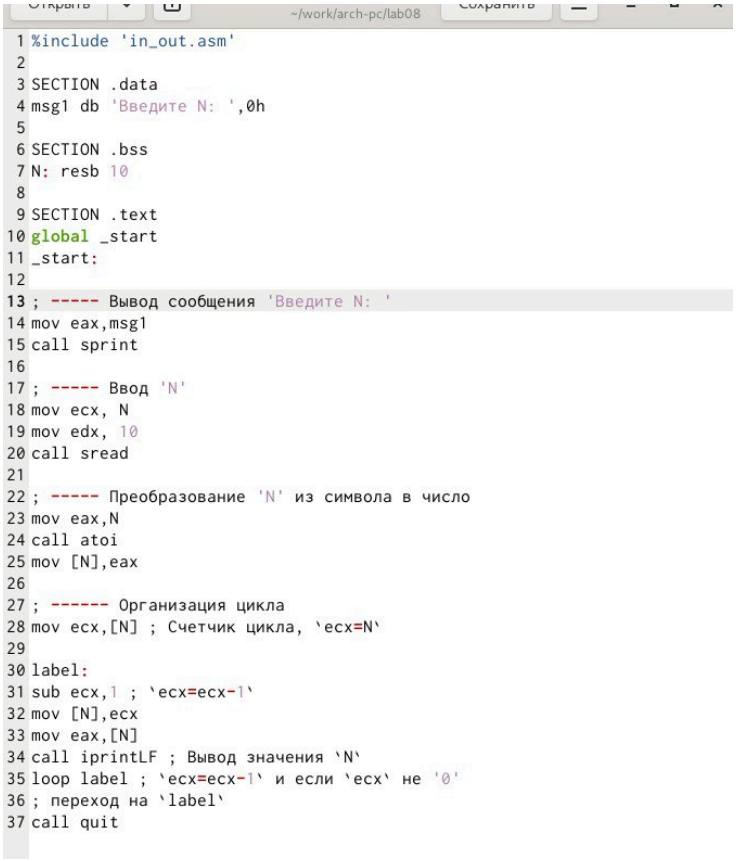

Рис. 4 Изменение программы

Из-за того, что теперь регистр ecx на каждой итерации уменьшается на 2 значения, количество итераций уменьшается вдвое. (рис. 5).

```
azsilantjeva@dk6n13 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
azsilantjeva@dk6n13 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
azsilantjeva@dk6n13 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 10
9
7
5
3
1
azsilantjeva@dk6n13 ~/work/arch-pc/lab08 $
```

Рис. 5 Запуск измененной программы

Добавляю команды push и pop в программу. (рис. 6)

Открыть | lab8-1.asm ~/work/arch-pc/lab08 | Сохранить

```
%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
```

Рис. 6 Добавление push и pop в цикл программы

Теперь количество итераций совпадает введенному N, но произошло смещение выводимых чисел на -1 (рис. 7).

```
azsilantjeva@dk6n13 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
azsilantjeva@dk6n13 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
azsilantjeva@dk6n13 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 10
9
8
7
6
5
4
3
2
1
0
azsilantjeva@dk6n13 ~/work/arch-pc/lab08 $
```

Рис. 7 Запуск измененной программы

2.2 Обработка аргументов командной строки

Создаю новый рабочий файл и вставляю в него код из следующего листинга (рис. 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. 8 Копирование программы из листинга

Запускаю вторую программу. (рис. 9)

Рис. 9 Запуск второй программы

Создаю новый файл для программы и копирую в него код из третьего листинга (рис. 10).

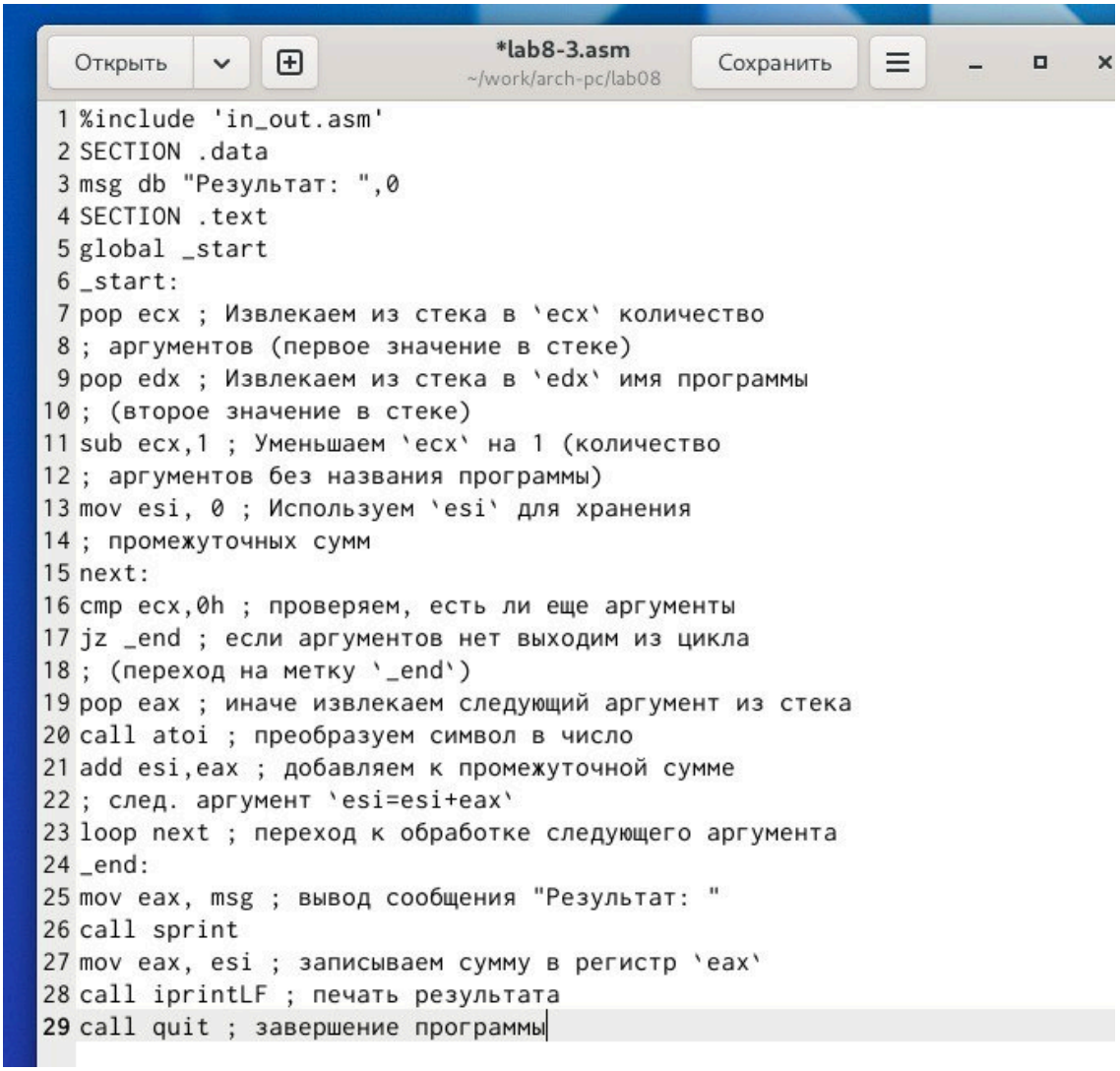

Рис. 10 Копирование программы из третьего листинга

Компилирую программу и запускаю ее, указав в качестве аргументов некоторые числа, которые программа складывает (рис. 11).

```
azsilantjeva@dk6n13 ~/work/arch-pc/lab08 $ touch lab8-3.asm
azsilantjeva@dk6n13 ~/work/arch-pc/lab08 $ nasm -f elf lab8-3.asm
azsilantjeva@dk6n13 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-3 lab8-3.o
azsilantjeva@dk6n13 ~/work/arch-pc/lab08 $ ./lab8-3 12 13 7 10 5
Результат: 47
azsilantjeva@dk6n13 ~/work/arch-pc/lab08 $
```

Рис. 11 Запуск третьей программы

Изменяю поведение программы так, чтобы указанные аргументы она умножала, а не складывала (рис. 12).

Открыть | lab8-3.asm ~/work/arch-pc/lab08 | Сохранить

```
1  %include 'in_out.asm'
2
3  SECTION .data
4  msg db "Произведение: ", 0  ; Сообщение для вывода
5  SECTION .text
6  global _start
7
8  _start:
9      pop ecx                 ; Извлекаем из стека количество аргументов
10     pop edx                 ; Извлекаем имя программы
11
12     sub ecx, 1              ; Уменьшаем ecx на 1 (количество аргументов
   без названия программы)
13
14     mov esi, 1              ; Используем esi для хранения произведения
   (начальное значение 1)
15
16 next:
17     cmp ecx, 0              ; Проверяем, есть ли еще аргументы
18     jz _end                 ; Если аргументов нет, выходим из цикла
19
20     pop eax                 ; Извлекаем следующий аргумент из стека
21     call atoi               ; Преобразуем строку в число
22     imul esi, eax           ; Умножаем промежуточное произведение на
   текущее значение
23
24     loop next               ; Переход к обработке следующего аргумента
25
26 _end:
27     mov eax, msg            ; Вывод сообщения "Произведение: "
28     call sprint
29
30     mov eax, esi            ; Записываем произведение в регистр eax
31     call iprintLF           ; Печать результата
32
33     call quit               ; Завершение программы
```

Рис. 12 Изменение третьей программы

Программа действительно теперь умножает данные на вход числа (рис. 13).

Рис. 13 Запуск измененной третьей программы

3. Задание для самостоятельной работы

Пишу программу, которая будет находить сумма значений для функции f(x) = 3x-1, которая совпадает с моим 2 вариантом. (рис.14)

Открыть | samost-1.asm ~/Загрузки | Сохранить

```
%include 'in_out.asm'

SECTION .data
msg_func db "Функция: f(x) = 3x - 1", 0
msg_result db "Результат: ", 0

SECTION .text
GLOBAL _start

_start:
mov eax, msg_func
call sprintLF

pop ecx
pop edx
sub ecx, 1
mov esi, 0

next:
cmp ecx, 0h
jz _end
pop eax
call atoi

mov ebx, 3
mul ebx
sub eax, 1

add esi, eax

loop next

_end:
mov eax, msg_result
call sprint
mov eax, esi
call iprintLF
call quit

```

Рис. 14 Написание программы для самостоятельной работы

Проверяю работу программы, указав в качестве аргумента несколько чисел (рис. 15).

```
azsilantjeva@dk6n08 ~/work/arch-pc/lab08 $ touch samost-1.asm
azsilantjeva@dk6n08 ~/work/arch-pc/lab08 $ nasm -f elf samost-1.asm
azsilantjeva@dk6n08 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o samost-1 samost-1.o
azsilantjeva@dk6n08 ~/work/arch-pc/lab08 $ ./samost-1123
bash: ./samost-1123: Нет такого файла или каталога
azsilantjeva@dk6n08 ~/work/arch-pc/lab08 $ ./samost-1 1 2 3
Функция: f(x) = 3x - 1
Результат: 15
azsilantjeva@dk6n08 ~/work/arch-pc/lab08 $
```

Рис. 15 Запуск программы для самостоятельной работы

3. Вывод

В результате выполнения данной лабораторной работы я приобрел навыки написания программ с использованием циклов а также научился обрабатывать аргументы командной строки.

Список литературы

1. GDB: The GNU Project Debugger. — URL: https://www.gnu.org/software/gdb/.
2. GNU Bash Manual. — 2016. — URL: https://www.gnu.org/software/bash/manual/.
3. Midnight Commander Development Center. — 2021. — URL: https://midnight-commander.org/.
4. NASM Assembly Language Tutorials. — 2021. — URL: https://asmtutor.com/.
5. Newham C. Learning the bash Shell: Unix Shell Programming. — O'Reilly Media, 2005. —
6. 354 c. — (In a Nutshell). — ISBN 0596009658. — URL: http://www.amazon.com/Learningbash- Shell-Programming-Nutshell/dp/0596009658.
7. Robbins A. Bash Pocket Reference. — O'Reilly Media, 2016. — 156 c. — ISBN 978-1491941591.
8. The NASM documentation. — 2021. — URL: https://www.nasm.us/docs.php.
9. Zarrelli G. Mastering Bash. — Packt Publishing, 2017. — 502 c. — ISBN 9781784396879.
10. Колдаев В. Д., Лупин С. А. Архитектура ЭВМ. — М. : Форум, 2018.
11. Куляс О. Л., Никитин К. А. Курс программирования на ASSEMBLER. — М. : Солон-Пресс, 2017.
12. Новожилов О. П. Архитектура ЭВМ и систем. — М. : Юрайт, 2016.
13. Расширенный ассемблер: NASM. — 2021. — URL: https://www.opennet.ru/docs/RUS/nasm/.
14. Робачевский А., Немнюгин С., Стесик О. Операционная система UNIX. — 2-е изд. — БХВПетербург, 2010. — 656 с. — ISBN 978-5-94157-538-1.
15. Столяров А. Программирование на языке ассемблера NASM для ОС Unix. — 2-е изд. — М. : МАКС Пресс, 2011. — URL: http://www.stolyarov.info/books/asm_unix.
16. Таненбаум Э. Архитектура компьютера. — 6-е изд. — СПб. : Питер, 2013. — 874 с. — (Классика Computer Science).
17. Таненбаум Э., Бос Х. Современные операционные системы. — 4-е изд. — СПб. : Питер, 2015. — 1120 с. — (Классика Computer Science).