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: Володина Алиса Алексеевна

Группа: НКАбд-01-25

МОСКВА

2025 г.

Оглавление

Список иллюстраций

1 Цель работы

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

2 Задания

1. Реализация циклом в NASM

2. Обработка аргументов командной строки

3. Самостоятельное написание программы по материалам лабораторной работы

3 Теоретическое введение

Стек — это структура данных, организованная по принципу LIFO («Last In — First Out» или «последним пришёл—первым ушёл»). Стек является частью архитектуры процессора и реализована аппаратном уровне. Для работы со стеком в процессоре есть специальные регистры (ss,bp,sp) и команды. Основной функцией стека является функция сохранения адресов возврата и передачи аргументов при вызове процедур. Кроме того, в нём выделяется память для локальных переменных и могут временно храниться значения регистров

4 Выполнение работы

Реализация циклов в NASM

Создадим каталог для программам лабораторной работы №8, перейдем в него и создадим файл lab8-1.asm (рисунок 0.1-0.3)

Рисунок 0.1 создание каталога

Рисунок 0.2 переход в каталог

Рисунок 0.3 создание файла lab8-1.asm

Введем в файл lab8-1.asm текст программы из листинга(рисунок 0.4)

```
%include 'in_out.asm'

SECTION .data
msg1 db 'Введите N: ',0h

SECTION .bss
N: resb 10

SECTION .text
global _start
_start:

mov eax,msg1
call sprint

mov ecx, N
mov edx, 10
call sread

mov eax,N
call atoi
mov [N],eax

mov ecx,[N]
label:
mov [N],ecx
mov eax,[N]
call iprintLF
loop label

call quit
```

Рисунок 0.4 запись программы

Создадим исполняемый файл и проверим его работу (рисунок 0.5-0.8)

```
aavolodina@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
aavolodina@fedora:~/work/arch-pc/lab08$
```

Рисунок 0.5 создание исполняемого файла

```
aavolodina@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
aavolodina@fedora:~/work/arch-pc/lab08$
```

Рисунок 0.6 создание исполняемого файла

```
aavolodina@fedora:~/work/arch-pc/lab08$ ./lab8-1
Введите N:
```

Рисунок 0.7 запуск исполняемого файла

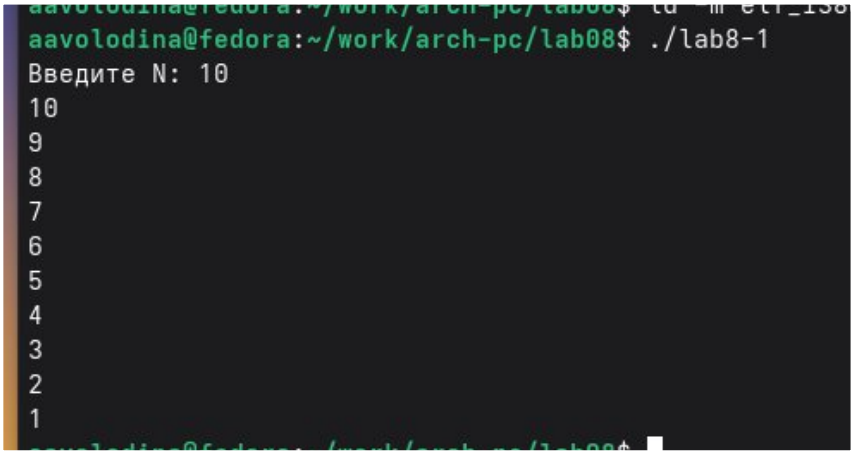

Рисунок 0.8 проверка работы программы

Изменим текст программы добавив изменение значения регистра ecx в цикле (рисунок 0.9-0.12)

```
%include 'in_out.asm'

SECTION .data
msg1 db 'Введите N: ',0h

SECTION .bss
N: resb 10

SECTION .text
global _start
_start:

mov eax,msg1
call sprint

mov ecx, N
mov edx, 10
call sread

mov eax,N
call atoi
mov [N],eax

mov ecx,[N]
label:
sub ecx,1 ; `ecx=ecx-1`
mov [N],ecx
mov eax,[N]
call iprintLF
loop label

call quit
```

Рисунок 0.9 изменение программы

Создадим исполняемый файл и проверьте его работу

Рисунок 0.10 создание исполняемого файла

Рисунок 0.11 запуск исполняемого файла

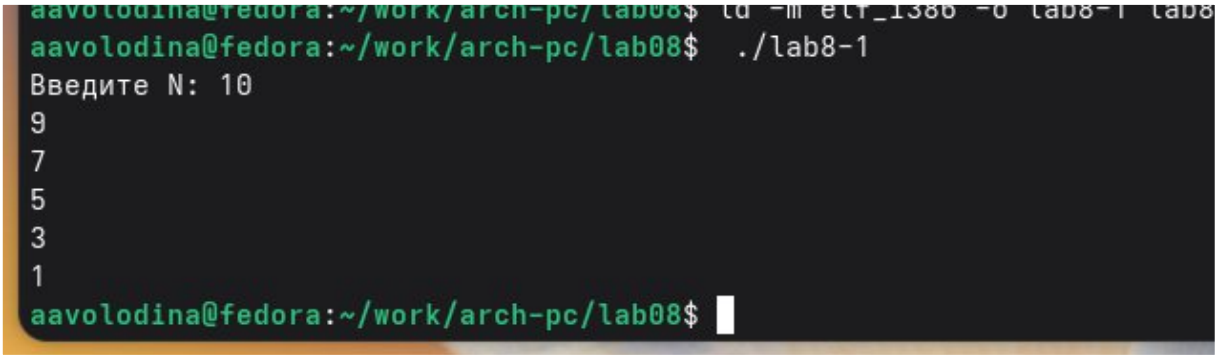

Рисунок 0.12 проверка работы программы

Количество итераций уменьшается вдвое ввиду того, что регистр ecx на каждой итерации стал меньше на 2 значения

Внесем изменения в текст программы, добавив команды push и pop для сохранения значения счетчика цикла loop (рисунок 0.13)

```
%include 'in_out.asm'

SECTION .data
msg1 db 'Введите N: ',0h

SECTION .bss
N: resb 10

SECTION .text
global _start
_start:

mov eax,msg1
call sprint

mov ecx, N
mov edx, 10
call sread

mov eax,N
call atoi
mov [N],eax

mov ecx,[N]
label:
push ecx
sub ecx,1
mov [N],ecx
mov eax,[N]
call iprintLF
pop ecx
loop label

call quit
```

Рисунок 0.13 изменение программы

Создадим исполняемый файл и проверим его работу (рисунок 0.14-0.16)

```
aavolodina@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
aavolodina@fedora:~/work/arch-pc/lab08$
```

Рисунок 0.14 создание исполняемого файла

```
aavolodina@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
aavolodina@fedora:~/work/arch-pc/lab08$
```

Рисунок 0.15 создание исполняемого файла

```
aavolodina@fedora:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 10
9
8
7
6
5
4
3
2
1
0
aavolodina@fedora:~/work/arch-pc/lab08$
```

Рисунок 0.16 проверка работы программы

Произошло смещение выводимых чисел на -1, но теперь количество итераций совпадает введенному N

Обработка аргументов командной строки

Создадим файл lab8-2.asm в каталоге ~/work/arch-pc/lab08 и введем в него текст программы из листинга (рисунок 0.17)

Открыть | *lab8-2.asm ~/work/arch-pc/lab08

```
1 %include 'in_out.asm'
2
3 SECTION .text
4 global _start
5
6 _start:
7 pop ecx
8 pop edx
9 sub ecx, 1
10
11 next:
12 cmp ecx, 0
13 jz _end
14 pop eax
15 call sprintLF
16 loop next
17
18 _end:
19 call quit
```

Рисунок 0.17 запись программы

Создадим исполняемый файл и запустим его, указав аргументы (рисунок

0.18-0.19)

```
aavolodina@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-2 lab8-2.o
aavolodina@fedora:~/work/arch-pc/lab08$
```

Рисунок 0.18 создание исполняемого файла

```
aavolodina@fedora:~/work/arch-pc/lab08$ ./lab8-2 аргумент1 аргумент 2 'аргумент 3'
аргумент1
аргумент
2
аргумент 3
aavolodina@fedora:~/work/arch-pc/lab08$
```

Рисунок 0.19 проверка работы программы

Программа обработала то же количество аргументов, что и было введено

Создадим файл lab8-3.asm в каталоге ~/work/arch pc/lab08 и введем в него текст программы из листинга (рисунок 0.20-0.21)

```
aavolodina@fedora:~/work/arch-pc/lab08$ touch lab8-3.asm
aavolodina@fedora:~/work/arch-pc/lab08$
```

Рисунок 0.20 создание файла

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

```
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
11 pop ecx
12
13 pop edx
14 sub ecx,1
15 mov esi, 0
16
17 next:
18 cmp ecx,0h
19 jz _end
20 pop eax
21 call atoi
22 add esi,eax
23
24 loop next
25 _end:
26 mov eax, msg
27 call sprint
28 mov eax, esi
29 call iprintLF
30 call quit
```

Рисунок 0.21 запись программы

Создадим исполняемый файл и запустим его, указав аргументы (рисунок 0.22-0.25)

```
aavolodina@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
aavolodina@fedora:~/work/arch-pc/lab08$
```

Рисунок 0.22 создание исполняемого файла

```
aavolodina@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
aavolodina@fedora:~/work/arch-pc/lab08$
```

Рисунок 0.23 создание исполняемого файла

```
aavolodina@fedora:~/work/arch-pc/lab08$ ./lab8-3 12 13 7 10 5
```

Рисунок 0.24 запуск исполняемого кода

```
aavolodina@fedora:~/work/arch-pc/lab08$ ./lab8-3 12 13 7 10 5
Результат: 47
aavolodina@fedora:~/work/arch-pc/lab08$
```

Рисунок 0.25 проверка работы программы

Изменим текст программы из листинга для вычисления произведения аргументов командной строки (рисунок 0.26-0.29)

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

```
%include 'in_out.asm'

SECTION .data
msg db "Результат: ", 0

SECTION .text
GLOBAL _start

_start:
pop ecx
pop edx
sub ecx, 1
mov esi, 1

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

loop next

_end:
mov eax, msg
call sprint
mov eax, esi
call iprintLF
call quit
```

Рисунок 0.26 изменение программы

```
aavolodina@fedora:~/work/arch-pc/lab08$ gedit lab8-3.asm
aavolodina@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
```

Рисунок 0.27 создание исполняемого файла

```
aavolodina@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
```

Рисунок 0.28 создание исполняемого файла

```
aavolodina@fedora:~/work/arch-pc/lab08$ ./lab8-3 12 13 7 10 5
Результат: 54600
```

Рисунок 0.29 проверка работы программы

Задание для самостоятельной работы

Напишем программу, которая находит сумму значений функции $f(x)$ для $x = x1, x2, ..., xn$, т.е. программа должна выводить значение $f(x1)+f(x2)+...+f(xn)$. Выражение для f(x)= $3x-1$. (рисунок 0.30)

Открыть

```
1 %include 'in_out.asm'
2
3 SECTION .data
4 msg_func db "Функция:f(x) = 3x-1", 0
5 msg_result db "Результат: ", 0
6
7 SECTION .text
8 GLOBAL _start
9
10 _start:
11 mov eax, msg_func
12 call sprintLF
13
14 pop ecx
15 pop edx
16 sub ecx, 1
17 mov esi, 0
18
19 next:
20 cmp ecx, 0h
21 jz _end
22 pop eax
23 call atoi
24
25 mov ebx, 3
26 mul ebx
27 sub eax, 1
28
29 add esi, eax
30
31 loop next
32
33 _end:
34 mov eax, msg_result
35 call sprint
36 mov eax, esi
37 call iprintLF
38 call quit
```

Рисунок 0.30 запись кода

Создадим исполняемый файл и проверьте его работу на нескольких наборах $x = x1,x2,...,xn$ (рисунок 0.31-0.34)

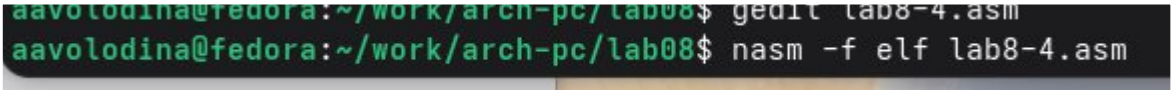

Рисунок 0.31 создание исполняемого файла

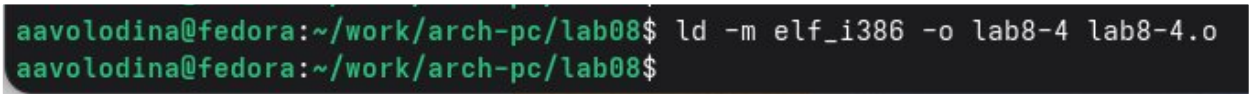

Рисунок 0.32 создание исполняемого файла

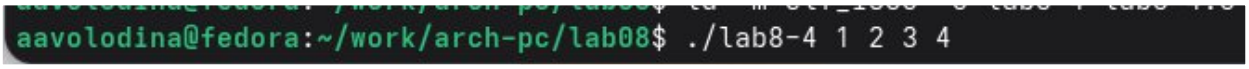

Рисунок 0.33 запуск исполняемого файла

Рисунок 0.34 проверка работы программы

5 Выводы

Я приобрела навыки написания программ с использованием циклов и обработкой аргументов командной строки.

Список литературы

1. GDB:TheGNUProjectDebugger.—URL:https://www.gnu.org/software/gdb/.

2. GNUBashManual.—2016.—URL:https://www.gnu.org/software/bash/manual/.

3. Midnight CommanderDevelopment Center.—2021.—URL: https://midnight-commander. org/.

4. NASMAssemblyLanguageTutorials.—2021.—URL: https://asmtutor.com/.

5. Newham C. Learning the bash Shell: Unix Shell Programming. — O'Reilly Media, 2005. — 354 c. —(In a Nutshell). —ISBN 0596009658.—URL: http://www.amazon.com/Learning bash-Shell-Programming-Nutshell/dp/0596009658.

6. RobbinsA. Bash Pocket Reference.—O'Reilly Media,2016.—156 c.—ISBN 978-1491941591.

7. TheNASMdocumentation.—2021.—URL:https://www.nasm.us/docs.php.

8. Zarrelli G. Mastering Bash.—Packt Publishing,2017.—502 c.—ISBN 9781784396879.

9. Колдаев В. Д.,Лупин С. А.Архитектура ЭВМ.—М. : Форум,2018.

10. Куляс О. Л.,Никитин К. А. Курс программирования наASSEMBLER.—М. : Солон-Пресс, 2017.

11. НовожиловО.П.Архитектура ЭВМисистем.—М.:Юрайт,2016.

12. Расширенныйассемблер:NASM.—2021.—URL:https://www.opennet.ru/docs/RUS/nasm/.

13. Робачевский А.,Немнюгин С.,Стесик О. Операционная система UNIX.—2-е изд.—БХВ Петербург, 2010.—656 с.—ISBN 978-5-94157-538-1.

14. СтоляровА.Программированиенаязыкеассемблера NASMдляОСUnix.—2-еизд.— М. : МАКСПресс,2011.—URL:http://www.stolyarov.info/books/asm_unix.

15. Таненбаум Э. Архитектура компьютера. — 6-е изд. — СПб. : Питер, 2013. — 874 с. — (Классика Computer Science).

16. Таненбаум Э.,Бос Х. Современные операционные системы.—4-е изд.—СПб.: Питер, 2015. —1120 с.—(Классика Computer Science)