

Отчёт по лабораторной работе №8

Дисциплина: архитектура компьютера

Назармамадов Умед Джамshedович

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
5	Вопросы для самопроверки	15
6	Выводы	16
	Список литературы	17

Список иллюстраций

4.1	Создание каталога и файла	8
4.2	Редактирование файла	8
4.3	Запуск файла	9
4.4	Редактирование файла	9
4.5	Запуск файла	10
4.6	Редактирование файла	11
4.7	Создание файла	11
4.8	Создание файла	11
4.9	Редактирование файла	12
4.10	Запуск программы	12
4.11	Создание файла	13
4.12	Запуск файла	13
4.13	Редактирование файла	14
4.14	Запуск файла	14

Список таблиц

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. Вопросы для самопроверки.

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 реализован на 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. Стек имеет вершину, адрес последнего добавленного элемента, который хранится в регистре esp (указатель стека). Противоположный конец стека называется дном. Значение, помещённое в стек последним, извлекается первым. При помещении значения в стек указатель стека уменьшается, а при извлечении — увеличивается. Для стека существует две основные операции: • добавление элемента в вершину стека (push); • извлечение элемента из вершины стека (pop). Команда push размещает значение в стеке, т.е. помещает значение в ячейку памяти, на которую указывает регистр esp, после этого значение регистра esp увеличивается на 4. Команда pop извлекает значение из стека, т.е. извлекает значение из ячейки памяти, на которую указывает регистр esp, после этого уменьшает значение регистра esp на 4. У этой команды также один операнд, который может быть регистром или переменной в памяти.

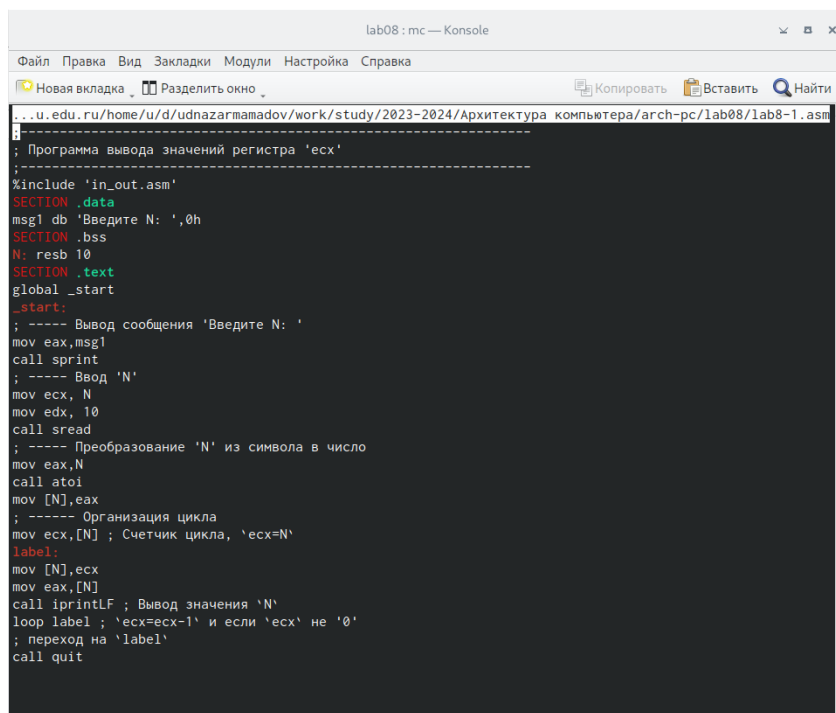
4 Выполнение лабораторной работы

Создаю каталог для программ лабораторной работы No 8, перехожу в него и создаю файл lab8-1.asm (рис. 4.14).

```
udnazarmamadov@dk5n56 ~ $ mkdir ~/work/study/2023-2024/"Архитектура компьютера"/arch-pc/lab08
udnazarmamadov@dk5n56 ~ $ cd ~/work/study/2023-2024/"Архитектура компьютера"/arch-pc/lab08
udnazarmamadov@dk5n56 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08 $ touch lab8-1.asm
```

Рис. 4.1: Создание каталога и файла

Захожу в файл lab8-1.asm, и ввожу туда текст программы из листинга 8.1 (рис. 4.14).



```
lab08 : mc — Konsole
Файл Правка Вид Закладки Модули Настройка Справка
Новая вкладка Разделить окно Копировать Вставить Найти
...u.edu.ru/home/u/d/udnazarmamadov/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08/lab8-1.asm
; Программа вывода значений регистра 'ecx'
;-----
%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:
mov [N],ecx
mov eax,[N]
call iprintLF ; Вывод значения 'N'
loop label ; 'ecx=ecx-1' и если 'ecx' не '0'
; переход на 'label'
call quit
```

Рис. 4.2: Редактирование файла

Создаю исполняемый файл и запускаю его (рис. 4.14).

```
udnazarmamadov@dk5n56 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08 $ nasm -f elf lab8-1.asm
udnazarmamadov@dk5n56 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
udnazarmamadov@dk5n56 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08 $ ./lab8-1
Введите N: 23
23
22
21
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
```

Рис. 4.3: Запуск файла

Изменяю текст программы, добавив изменение значения регистра ecx в цикле (рис. 4.14).

```
/afs/.dk.sei.pfu.edu.ru/home/u/d/udnazarmamadov/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08/lab8-1.asm
;-----
; Программа вывода значений регистра 'ecx'
;-----
%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:
sub ecx,1 ; 'ecx=ecx-1'
mov [N],ecx
mov eax,[N]
call iprintfLF
loop label
call quit
```

Рис. 4.4: Редактирование файла

Запускаю файл для проверки (рис. 4.14).

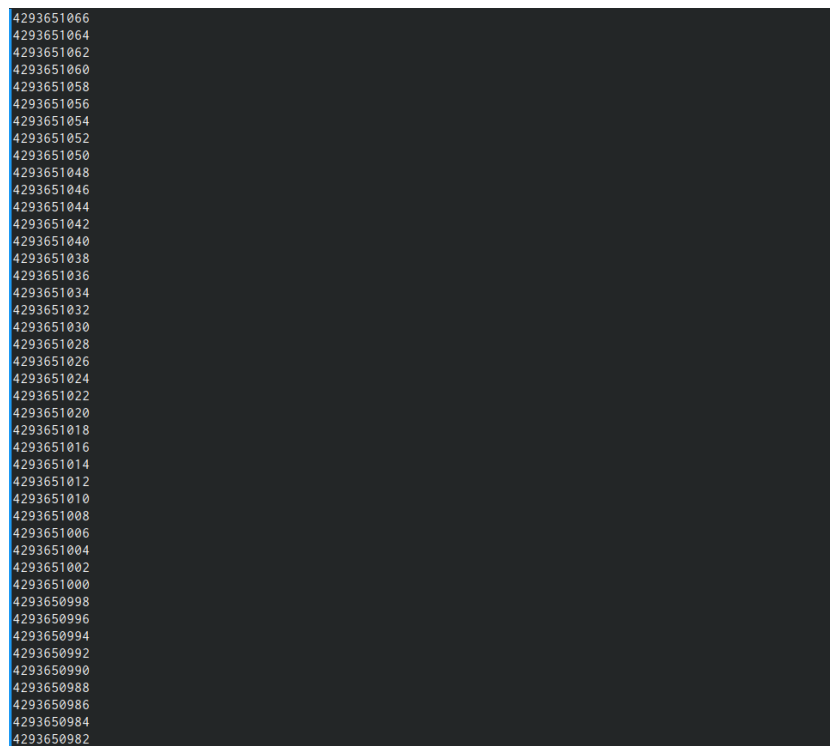


Рис. 4.5: Запуск файла

Число проходов цикла не соответствует введенному с клавиатуры значению.

Вношу изменения в программу, добавив команды push и pop для сохранения значения счетчика цикла loop (рис. 4.14).

```

...ci.pfu.edu.ru/home/u/d/udnazarmamadov/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08/lab8-1.asm Изменён
;-----
; Программа вывода значений регистра 'ecx'
;-----
#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 iprintf
call iprintfLF
pop ecx ; извлечение значения ecx из стека

loop label
call quit

```

Рис. 4.6: Редактирование файла

Создаю исполняемый файл и проверяю его работу (рис. 4.14).

```

udnazarmamadov@dk5n56 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08 $ nasm -f elf lab8-1.asm
udnazarmamadov@dk5n56 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
udnazarmamadov@dk5n56 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08 $ ./lab8-1
Введите N: 3
2
1
0

```

Рис. 4.7: Создание файла

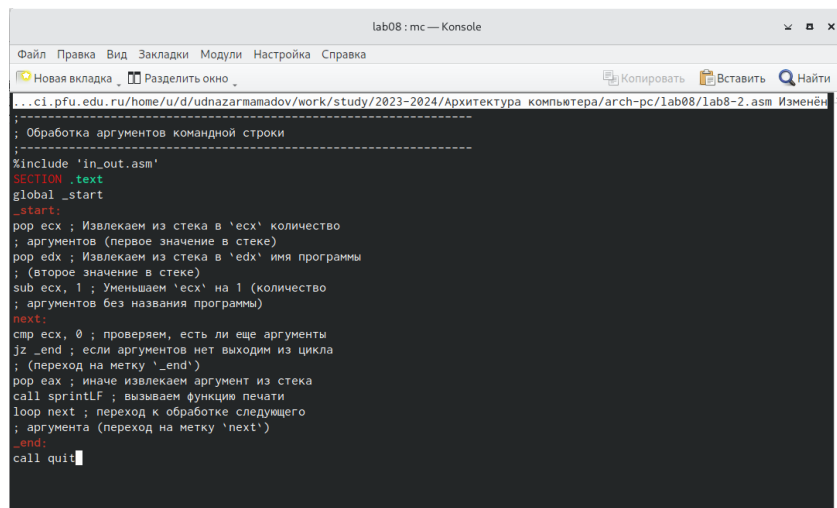
Создаю файл lab8-2.asm в каталоге ~/work/study/2023-2024/“Архитектура компьютера”/arch-pc/lab08 и ввожу в него текст программы из листинга 8.2 (рис. 4.14).

```

udnazarmamadov@dk5n56 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08 $ touch lab8-2.asm

```

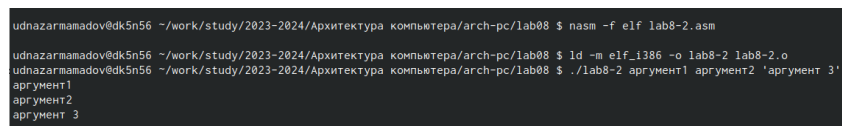
Рис. 4.8: Создание файла



```
lab08: mc — Konsole
Файл Правка Вид Закладки Модули Настройка Справка
Новая вкладка Разделить окно
...ci.pfu.edu.ru/home/u/d/udnazarmamadov/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08/lab8-2.asm Изменён
; Обработка аргументов командной строки
;-----
%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 sprintf ; вызываем функцию печати
    loop next ; переход к обработке следующего
             ; аргумента (переход на метку 'next')
_end:
    call quit
```

Рис. 4.9: Редактирование файла

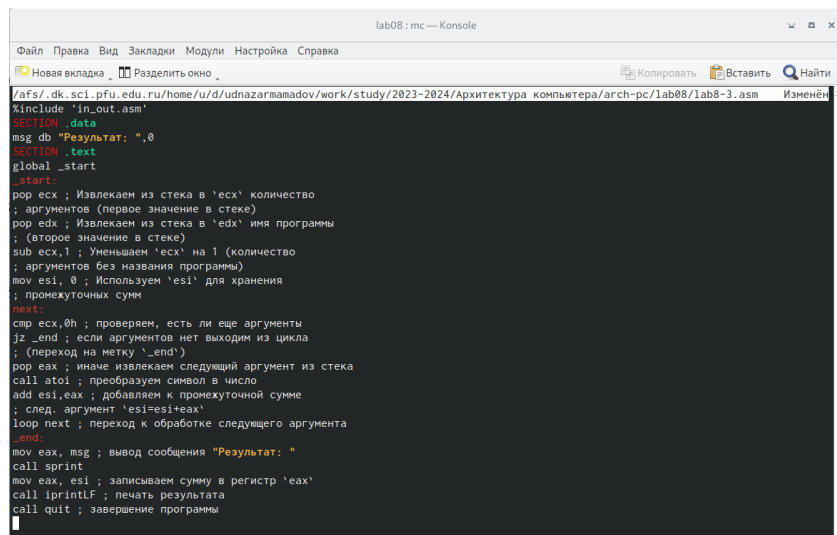
Создаю исполняемый файл и запускаю его (рис. 4.14).



```
udnazarmamadov@dk5n56 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08 $ nasm -f elf lab8-2.asm
udnazarmamadov@dk5n56 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08 $ ld -m elf_i386 -o lab8-2 lab8-2.o
udnazarmamadov@dk5n56 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08 $ ./lab8-2 аргумент1 аргумент2 'аргумент 3'
аргумент1
аргумент2
аргумент 3
```

Рис. 4.10: Запуск программы

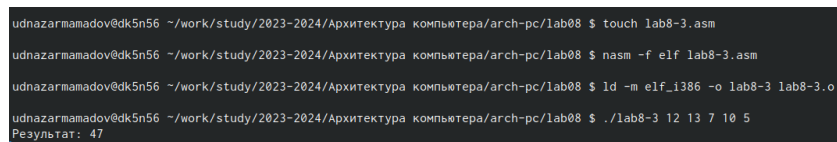
Создаю еще один файл lab8-3.asm, которая выводит сумму чисел, которые передаются в программу как аргументы (рис. 4.14).



```
lab08: mc — Konsole
Файл Правка Вид Закладки Модули Настройка Справка
Новая вкладка Разделить окно Копировать Вставить Найти
/afs/dk.sci.pfu.edu.ru/home/u/d/udnazarmamadov/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08/lab8-3.asm Изменить
#include "in_out.asm"
SECTION .data
msg db "Результат: ",0
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)
    mov esi, 0 ; Используем 'esi' для хранения
    ; промежуточных сумм
next:
    cmp ecx,0h ; проверяем, есть ли еще аргументы
    jz _end ; если аргументов нет выходим из цикла
    ; (переход на метку '_end')
    pop eax ; иначе извлекаем следующий аргумент из стека
    call atoi ; преобразуем символ в число
    add esi,eax ; добавляем к промежуточной сумме
    ; след. аргумент 'esi=esi+eax'
    loop next ; переход к обработке следующего аргумента
_end:
    mov eax, msg ; вывод сообщения "Результат: "
    call sprintf
    mov eax, esi ; записываем сумму в регистр 'eax'
    call iprintf ; печать результата
    call quit ; завершение программы
```

Рис. 4.11: Создание файла

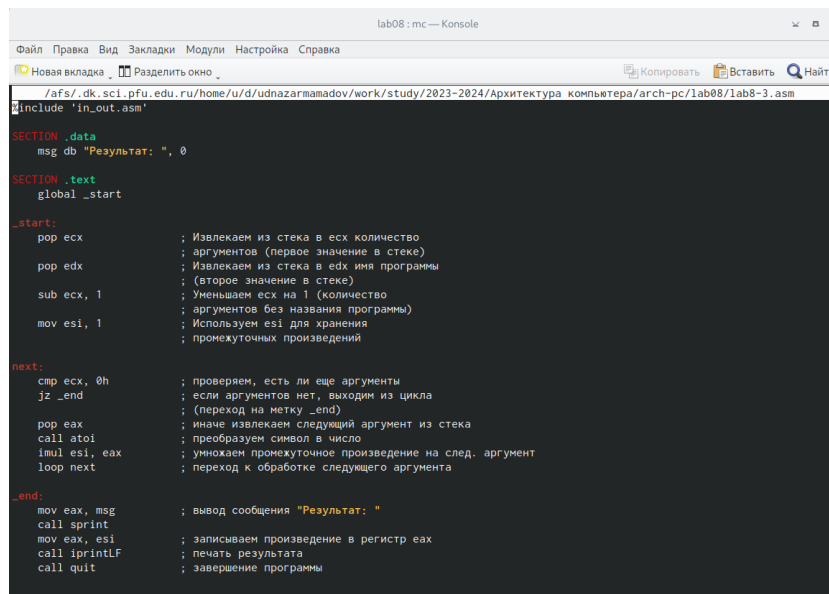
Создаю исполняемый файл и запускаю его (рис. 4.14).



```
udnazarmamadov@dk5n56 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08 $ touch lab8-3.asm
udnazarmamadov@dk5n56 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08 $ nasm -f elf lab8-3.asm
udnazarmamadov@dk5n56 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08 $ ld -m elf_i386 -o lab8-3 lab8-3.o
udnazarmamadov@dk5n56 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08 $ ./lab8-3 12 13 7 10 5
Результат: 47
```

Рис. 4.12: Запуск файла

Изменяю текст программы из листинга 8.3 для вычисления произведения аргументов командной строки (рис. 4.14).



```
lab08: mc — Konsole
Файл Правка Вид Закладки Модули Настройка Справка
Новая вкладка Разделить окно Копировать Вставить Найти
/afs/.dk.sci.pfu.edu.ru/home/u/d/udnazarmamadov/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08/lab8-3.asm
#include 'in_out.asm'

SECTION .data
msg db "Результат: ", 0

SECTION .text
global _start

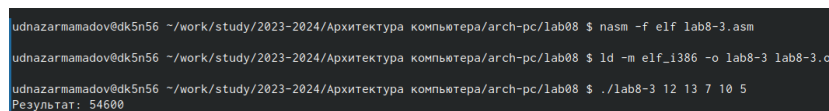
_start:
    pop ecx          ; Извлекаем из стека в ecx количество
                    ; аргументов (первое значение в стеке)
    pop edx          ; Извлекаем из стека в edx имя программы
                    ; (второе значение в стеке)
    sub ecx, 1        ; Уменьшаем ecx на 1 (количество
                    ; аргументов без названия программы)
    mov esi, 1        ; Используем esi для хранения
                    ; промежуточных произведений

next:
    cmp ecx, 0h      ; проверяем, есть ли еще аргументы
    jz _end          ; если аргументов нет, выходим из цикла
                    ; (переход на метку _end)
    pop eax          ; иначе извлекаем следующий аргумент из стека
    call atoi        ; преобразуем символ в число
    imul esi, eax     ; умножаем промежуточное произведение на след. аргумент
    loop next        ; переход к обработке следующего аргумента

_end:
    mov eax, msg      ; вывод сообщения "Результат: "
    call sprintf
    mov eax, esi      ; записываем произведение в регистр eax
    call iprintf
    call quit        ; завершение программы
```

Рис. 4.13: Редактирование файла

Создаю исполняемый файл и запускаю его (рис. 4.14).



```
udnazarmamadov@dk5n56 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08 $ nasm -f elf lab8-3.asm
udnazarmamadov@dk5n56 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08 $ ld -m elf_i386 -o lab8-3 lab8-3.o
udnazarmamadov@dk5n56 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08 $ ./lab8-3 12 13 7 10 5
Результат: 54600
```

Рис. 4.14: Запуск файла

5 Вопросы для самопроверки

1. Опишите работу команды `loop`. Проверка условия: Происходит автоматическое сравнение `ecx` с нулем. Если `ecx` не равен нулю, то выполнение программы переходит к следующему шагу. В противном случае, цикл завершается, и выполнение программы продолжается со следующей инструкции после `loop`. Переход к метке: Если условие (`ecx ≠ 0`) выполнено, происходит безусловный переход к метке, указанной в `destination`. Уменьшение счетчика цикла: После выполнения перехода, регистр `ecx` уменьшается на 1 (`ecx = ecx - 1`).
2. Как организовать цикл с помощью команд условных переходов, не прибегая к специальным командам управления циклами? В NASM можно организовать цикл с использованием команд условных переходов, таких как `jmp`, `je` (jump if equal), `jne` (jump if not equal), `jl` (jump if less), `jg` (jump if greater), и других.
3. Дайте определение понятия «стек». Стек — это структура данных, организованная по принципу LIFO («Last In — First Out» или «последним пришёл — первым ушёл»). Стек является частью архитектуры процессора и реализован на аппаратном уровне.
4. Как осуществляется порядок выборки содержащихся в стеке данных? Для стека существует две основные операции: • добавление элемента в вершину стека (`push`); • извлечение элемента из вершины стека (`pop`).

6 Выводы

При выполнения данной лабораторной работы я приобрел навыки написания программ с использованием циклов и обработкой аргументов 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. — 354 с. — (In a Nutshell). — ISBN 0596009658. — URL: <http://www.amazon.com/Learning-bash-Shell-Programming-Nutshell/dp/0596009658>.
6. Robbins A. Bash Pocket Reference. — O'Reilly Media, 2016. — 156 с. — ISBN 978-1491941591.
7. The NASM documentation. — 2021. — URL: <https://www.nasm.us/docs.php>.
8. Zarrelli G. Mastering Bash. — Packt Publishing, 2017. — 502 с. — ISBN 9781784396879.
9. Колдаев В. Д., Лупин С. А. Архитектура ЭВМ. — М. : Форум, 2018.
10. Куляс О. Л., Никитин К. А. Курс программирования на ASSEMBLER. — М. : Солон-Пресс,
- 11.
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. — БХВ- Петербург, 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,
18. — 1120 с. — (Классика Computer Science).