

# **Отчет по лабораторной работе №8**

**Дисциплина: архитектура компьютера**

Провоторов Антон Григорьевич

# Содержание

1	Цель работы	4
2	Теоретическое введение	5
3	Задание	6
4	Выполнение лабораторной работы	7
5	Выводы	13
6	Список литературы	14

## Список иллюстраций

4.1	Создание каталога lab08, переход в него и создание файла lab8-1.asm	7
4.2	Ввод текста из листинга в файла, создание исполняемого файла и проверка его работы . . . . .	7
4.3	Изменение текста в файле . . . . .	8
4.4	Проверка работы файла . . . . .	8
4.5	Изменение текста файла . . . . .	9
4.6	Проверка работы файла . . . . .	9
4.7	Ввод текста из листинга в файл . . . . .	10
4.8	Запуск исполняемого файла . . . . .	10
4.9	Ввод текста из листинга в файла . . . . .	11
4.10	Изменение текста в файле . . . . .	11
4.11	Текст программы . . . . .	12
4.12	Работа самостоятельной работы . . . . .	12

# 1 Цель работы

Приобретение навыков написания программ с использованием циклов и обработкой аргументов строки

## 2 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.

## **3 Задание**

Выполнить лабораторную и самостоятельную работу, и написать отчет на основании проделанной работы

## 4 Выполнение лабораторной работы

Создал каталог для программ лабораторной работы №8, перешел и создал файл lab8-1.asm (рис. 4.1).

```
agprovotorov@dk8n72 ~ $ mkdir /afs/.dk.sci.pfu.edu.ru/home/a/g/agprovotorov/work/arch-ps/lab08
agprovotorov@dk8n72 ~ $ cd /afs/.dk.sci.pfu.edu.ru/home/a/g/agprovotorov/work/arch-ps/lab08
agprovotorov@dk8n72 ~/work/arch-ps/lab08 $ touch lab8-1.asm
```

Рис. 4.1: Создание каталога lab08, переход в него и создание файла lab8-1.asm

Ввел в файл lab8-1.asm текст программы из листинга 8.1. Создал исполняемый файл и проверил его работу.(рис. 4.2).

```
agprovotorov@dk4n69 ~/work/arch-ps/lab08 $ nasm -f elf lab8-1.asm
agprovotorov@dk4n69 ~/work/arch-ps/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
agprovotorov@dk4n69 ~/work/arch-ps/lab08 $ ./lab8-1
Введите N: 10
10
9
8
7
6
5
4
3
2
1
```

Рис. 4.2: Ввод текста из листинга в файла, создание исполняемого файла и проверка его работы

Изменил текст файла lab8-1.asm(рис. 4.3).

```

lab8-1.asm [~M--] 9 L: [ 1+22 23/ 29] *(482 / 646b) 0010 0x00A
#include "stdio.h"
SECTION ".text"
msg1 db "Введите N: ",0h
SECTION ".bss"
N: resb 10
SECTION ".text"
global _start
_start:
; ----- Вывод сообщения "Введите N: "
mov eax,msg1
call sprintf
; ----- Ввод 'N'
mov ecx,N
mov edx,10
call sread
; ----- Преобразование 'N' из символа в число
mov ecx,N
call atoi
mov [N],eax
; ----- Организация цикла
mov ecx,[N] ; Счетчик цикла, 'ВведитеN'
label:
sub ecx,1
mov [N],ecx
call printf ; Вывод значения 'N'
loop label ; 'loop' не 'exit' на '0'
; переход на 'label'
call quit

```

Рис. 4.3: Изменение текста в файле

Проверил работу файла после изменение текста файла(рис. 4.4).

```

agprovotorov@dk8n72 ~/work/arch-ps/lab08 $ nasm -f elf lab8-1.asm
agprovotorov@dk8n72 ~/work/arch-ps/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
agprovotorov@dk8n72 ~/work/arch-ps/lab08 $ ./lab8-1
Введите N: 10
9
7
5
3
1

```

Рис. 4.4: Проверка работы файла

Изменил текст файла lab8-1.asm(рис. 4.5).



```

lab8-1.asm      [-M--]  1 L:[ 1+27 28/ 31] *(567 / 666b) 0111 0x06F
#include "lab8-1.inc"
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
sub ecx,1
mov [N],ecx
mov eax,[N]
call iprintLF ; Вывод значения 'N'
pop ecx
loop label ; 'ecx=ecx-1' и если 'ecx' не '0'
; переход на 'label'
call quit

```

Рис. 4.5: Изменение текста файла

Проверил работу файла после изменение текста файла(рис. 4.6).

```

agprovotorov@dk8n72 ~/work/arch-ps/lab08 $ nasm -f elf lab8-1.asm
agprovotorov@dk8n72 ~/work/arch-ps/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
agprovotorov@dk8n72 ~/work/arch-ps/lab08 $ ./lab8-1
Введите N: 10
9
8
7
6
5
4
3
2
1
0

```

Рис. 4.6: Проверка работы файла

Создал файл lab8-2.asm в каталоге ~/work/arch-ps/lab08 и ввел в него текст программы из листинга 8.2.(рис. 4.7).

```

lab8-2.asm      [-M--]  9 L:[ 1+19 20/ 20] *(943 / 943b) <EOF>
#include "lab8-2.h"
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 printf ; вызываем функцию печати
    loop next ; переход к обработке следующего
              ; аргумента (переход на метку 'next')
_end:
    call quit

```

Рис. 4.7: Ввод текста из листинга в файл

Создал исполняемый файл и запустил его, указав аргументы 10 5 4(рис. 4.8).

```

agprovotorov@dk8n72 ~/work/arch-ps/lab08 $ nasm -f elf lab8-2.asm
agprovotorov@dk8n72 ~/work/arch-ps/lab08 $ ld -m elf_i386 -o lab8-2 lab8-2.o
agprovotorov@dk8n72 ~/work/arch-ps/lab08 $ ./lab8-2 10 5 4
10
5
4

```

Рис. 4.8: Запуск исполняемого файла

Создал файл lab8-3.asm в каталоге ~/work/arch- ps/lab08 и ввел в него текст программы из листинга 8.3.(рис. 4.9).

```

lab8-3.asm      [-M--] 45 L:[ 1+15 16/ 30] *(689 /1429b) 0010 0x00A
#include "lab03.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 sprint
mov ecx, esi ; записываем сумму в регистр 'ecx'
call iprintf ; печать результата
call quit ; завершение программы

```

Рис. 4.9: Ввод текста из листнга в файла

Изменил текст файла lab8-3.asm в соответствии с заданием(рис. 4.10).

```

pop eax ; иначе извлекаем следующий аргумент из стека
call atoi ; преобразуем символ в число
mov edx,eax
mov eax,esi
mul edx ; добавляем к промежуточной сумме
; след. аргумент 'esi=esi+eax'
mov esi,eax
loop next ; переход к обработке следующего аргумента
_end:

```

Рис. 4.10: Изменение текста в файле

Проверил работу программы.Программа работает корректно(рис. ??).

```

agprovotorov@dk8n72 ~/work/arch-ps/lab08 $ nasm -f elf lab8-3.asm
agprovotorov@dk8n72 ~/work/arch-ps/lab08 $ ld -m elf_i386 -o lab8-3 lab8-3.o
agprovotorov@dk8n72 ~/work/arch-ps/lab08 $ ./lab8-3 12 13 7 10 5
Результат: 54600

```

## Самостоятельная

работа Так как мой вариант 1, написал программу для суммы значений функции  $f(x)=2x+15$  от введённых значений (рис. 4.11).

```

1 %include 'in_out.asm'
2 SECTION .data
3 msg1 db "Функция: 2x + 15 ", 0
4 msg db "Результат: ", 0
5 SECTION .text
6 global _start
7 _start:
8 pop ecx ; Извлекаем из стека в 'ecx' количество
9           ; аргументов (первое значение в стеке)
10 pop edx ; Извлекаем из стека в 'edx' имя программы
11           ; (второе значение в стеке)
12 sub ecx, 1 ; Уменьшаем 'ecx' на 1 (количество
13           ; аргументов без названия программы)
14 mov esi, 0 ; Используем 'esi' для хранения
15           ; промежуточных сумм
16 next:
17 cmp ecx, 0h ; проверяем, есть ли еще аргументы
18 jz _end ; если аргументов нет выходим из цикла
19           ; (переход на метку '_end')
20 pop eax ; иначе извлекаем следующий аргумент из стека
21 call atoi ; преобразуем символ в число
22 mov edx, 2
23 mul edx
24 add esi, eax ; добавляем к промежуточной сумме
25           ; след. аргумент 'esi=esi+eax'
26 mov edx, 15
27 add esi, edx
28 loop next ; переход к обработке следующего аргумента
29 _end:
30 mov eax, msg1
31 call sprintLF
32 mov eax, msg ; вывод сообщения "Результат: "
33 call sprintLF
34 mov eax, esi ; записываем сумму в регистр 'eax'
35 call iprintLF ; печать результата
36 call quit ; завершение программы

```

Рис. 4.11: Текст программы

Скомпилировав файл, убедимся, что он работает корректно (рис. 4.12).

```

agprovotorov@dk8n72 ~/work/arch-ps/lab08 $ nasm -f elf lab8-4.asm
agprovotorov@dk8n72 ~/work/arch-ps/lab08 $ ld -m elf_i386 -o lab8-4 lab8-4.o
agprovotorov@dk8n72 ~/work/arch-ps/lab08 $ ./lab8-4 1 2 3 4
Функция: 2x + 15
Результат:
80

```

Рис. 4.12: Работа самостоятельной работы

## 5 Выводы

Я научился использовать писать циклы на языке ассемблера, а также получать информацию из командной строки.

## 6 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://midnightcommander.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, 2017.
11. Новожилов О. П. Архитектура ЭВМ и систем. — М. : Юрайт, 2016.
12. Расширенный ассемблер: 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. 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](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,

17. — 1120 с. — (Классика Computer Science)