

# 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: Мурылев Иван Валерьевич

Группа: НПИбд-03-25

МОСКВА

2025 г.

# **Содержание**

**Цель работы**

**Теоретическое введение**

**Выполнение работы**

**Вывод**

# Цель работы

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

## 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.

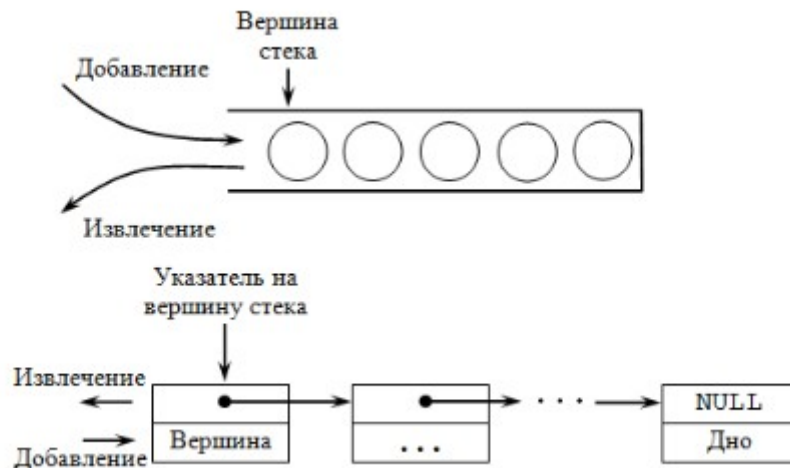
Стек имеет вершину, адрес последнего добавленного элемента, который хранится в регистре esp (указатель стека). Противоположный конец стека называется дном. Значение, помещённое в стек последним, извлекается первым. При помещении значения в стек указатель стека уменьшается, а при извлечении — увеличивается.

Для стека существует две основные операции:

- добавление элемента в вершину стека (push);
- извлечение элемента из вершины стека (pop)

Организация стека :

Ещё  
две



команды для добавления значений в стек. Это команда `pusha`, которая помещает в стек содержимое всех регистров общего назначения в следующем порядке: `ax`, `cx`, `dx`, `bx`, `sp`, `bp`, `si`, `di`. А также команда `pushf`, которая служит для перемещения в стек содержимого регистра флагов. Обе эти команды не имеют операндов.

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.

## Выполнение работы

### Лабораторная часть

Так как в структуре репозитория уже создана папка отчета, первый этап пропускается.

После создания файл lab08.asm и заполнения его литингом 8.1 операция cat выдаст:

```
/home/ivmurihlev/Documents/work/study/2025-2026/arch_Evm/arch-evm/labs/lab08/report  
[ivmurihlev@personal report]$ touch lab08.asm
```

```
%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
```

Затем транслируем и компанируем файл и получаем исполняемый файл lab08, который выдаст следующий результат:

```

[ivmurihlev@personal report]$ nasm -f elf lab08.asm
[ivmurihlev@personal report]$ ls
arch-pc--lab08--report.qmd  _assets  bib  image  in_out.asm  lab08.asm  lab08.o  Makefi
[ivmurihlev@personal report]$ ld -mb0elf_i386 lab08.o -o lab08  lab07-3.ls
[ivmurihlev@personal report]$ ./lab08/
bash: ./lab08/: Not a directory
[ivmurihlev@personal report]$ ls/
bash: ls/: No such file or directory
[ivmurihlev@personal report]$ ls
arch-pc--lab08--report.qmd  _assets  bib  image  in_out.asm  lab08  lab08.asm  lab08.o
[ivmurihlev@personal report]$ ./lab08
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

```

Как видно число итераций совпадает с числом N и вывод начинается с N и не доходит до 0.

После внесения изменения , предложенного в задании, в текст программы и создания исполняемого файла и его запуска получаем:

```
4291692942
4291692940
4291692938
4291692936
4291692934
4291692932
4291692930
4291692928
4291692926
4291692924
4291692922
4291692920
4291692918
4291692916
4291692914
4291692912
4291692910
4291692908
4291692906
4291692904
4291692902
4291692900
4291692898
4291692896
```

Как видно программа выдает бесконечную последовательность из случайных чисел (похоже на значение строк в листинге или случайное значение регистра).

После внесения второго изменения и запуска файла получаем:

```
[ivmurihlev@personal report]$ ./lab08
Введите N: 5
4
3
2
1
0
```

Теперь программа выводит 5 итераций считая от четырех, что напоминает вывод циклов в других более высоких языках программирования.

Во втором задании вносим текст листинга 8.2 в lab08-2.asm и создаем исполняемый файл и запускаем его с аргументами из примера.

```
[ivmurihlev@personal report]$ nasm -f elf lab08-2.asm
[ivmurihlev@personal report]$ ld -m elf_i386 lab08-2.o -o lab08-2
[ivmurihlev@personal report]$ ./lab08-2 аргумент1 аргумент 2 'аргумент 3'
аргумент1
аргумент
2
аргумент 3
```

Как видно программа опработала их по-разному из-за разного типа данных поступающих из стека к функции `sprintLF` из регистра `eax`. После создания файла `lab08-3.asm` и создания его исполняемого файла запускаем `lab08-3` с аргументами из примера.

```
[ivmurihlev@personal report]$ touch lab08-3.asm
[ivmurihlev@personal report]$ nasm -f elf lab08-3.asm
[ivmurihlev@personal report]$ ld -m elf_i386 lab08-3.o -o lab08-3
[ivmurihlev@personal report]$ ls
arch-pc--lab08--report.qmd  bib      in_out.asm  lab08-2      lab08-2.o  lab08-3.asm  lab08.asm  Makefile  _resources
_assets                    image     lab08       lab08-2.asm  lab08-3    lab08-3.o  lab08.o  quarto.yml
[ivmurihlev@personal report]$ ./main 12 13 7 10 5
bash: ./main: No such file or directory
[ivmurihlev@personal report]$ ./lab08-3 12 13 7 10 5
Результат: 47
```

## Самостоятельная часть

Создаю файл `lab08-4.asm` и пишу программу для 7 варианта, то есть  $f(x) = 3(x+2)$ . Затем делую из него исполняемый файл (Первые 3 строки). После запуска на трех наборах (`{3;3}`; `{3,3,5}`; `{1,2,3}`) получаем на скриншоте. После ручной проверки, подтверждаю корректность всех вариантов:

$$30 = (3*(3+2))*2 = 15*2$$

$$51 = 15*2 + 3*7 = 51$$

$$36 = 3*(1+2+2+2)+15$$

```
[ivmurihlev@personal report]$ touch lab08-4.asm
[ivmurihlev@personal report]$ nasm -f elf lab08-4.asm
[ivmurihlev@personal report]$ ld -m elf_i386 lab08-4.o -o samostoyatel'naya_chast
[ivmurihlev@personal report]$ ./samostoyatel'naya_chast 3 3
Функция 3(x+2)
Результат: 30
[ivmurihlev@personal report]$ ./samostoyatel'naya_chast 3 3 5
Функция 3(x+2)
Результат: 51
[ivmurihlev@personal report]$ ./samostoyatel'naya_chast 1 2 3
Функция 3(x+2)
Результат: 36
[ivmurihlev@personal report]$
```

## Вывод



Была изучена реализация механизма циклов на языке Assambler и структура стека в данном языке и была написана программа с использованием этого механизма. (списано что писать не надо)