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

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.

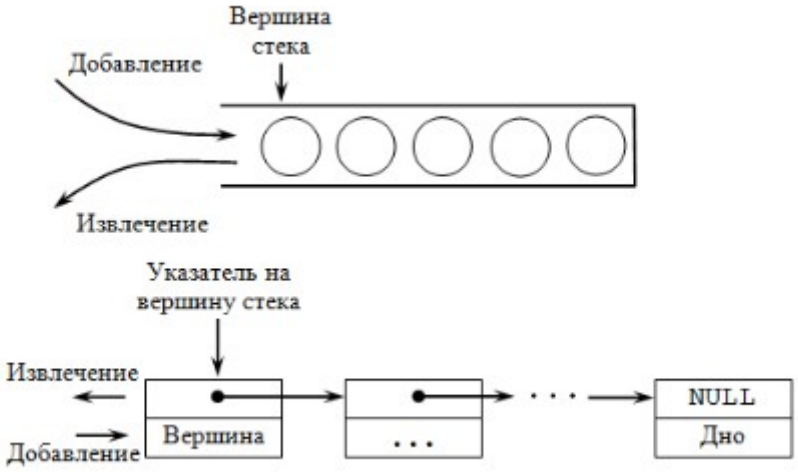
Для стека существует две основные операции:

• 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

Лабараторая часть

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

Копируем in-out.asm из папка lab05/report/ в lab08/report/.

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
```

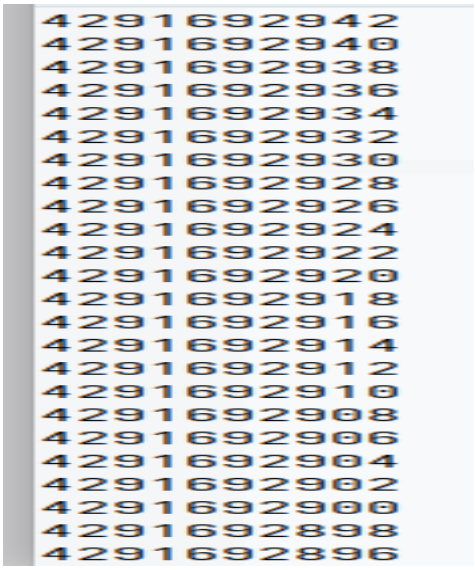
```
%include 'in_out.asm'
SECTION .data
msg1 db 'Введите N: ',0h
SECTION .bss
N: resb 10
SECTION .text
global _start
_start:
; ----- Вывод сообщения 'Введите N: '
mov eax,msg1
call sprint
; ----- Ввод 'N'
mov ecx, N
mov edx, 10
call sread
; ----- Преобразование 'N' из символа в число
mov eax,N
call atoi
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 компануем файл и получаем исполняемый файд 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 -m  elf_i386  lab08.o -o lab08
[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
```

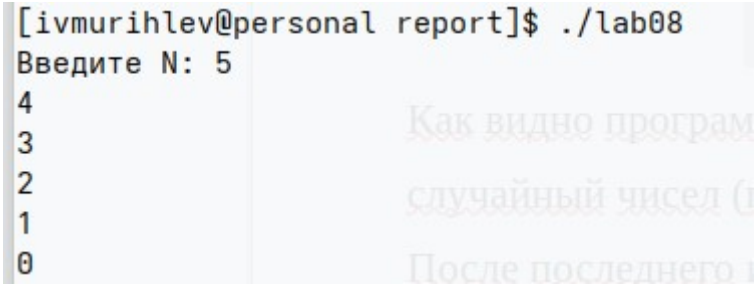
Как видно число итераций совпадает с числом N и вывод начинается с N и не доходит до 0.

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



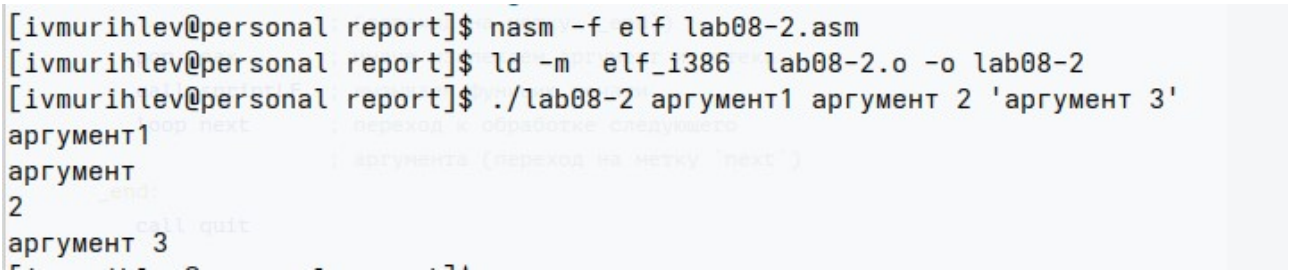
Как видно программа выдает бесконечную последовательность из случайный чисел (похоже на значение строк в листинге или случайное значение регистра).

После внесения второго изменения и запуска файла получаем:



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

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



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.

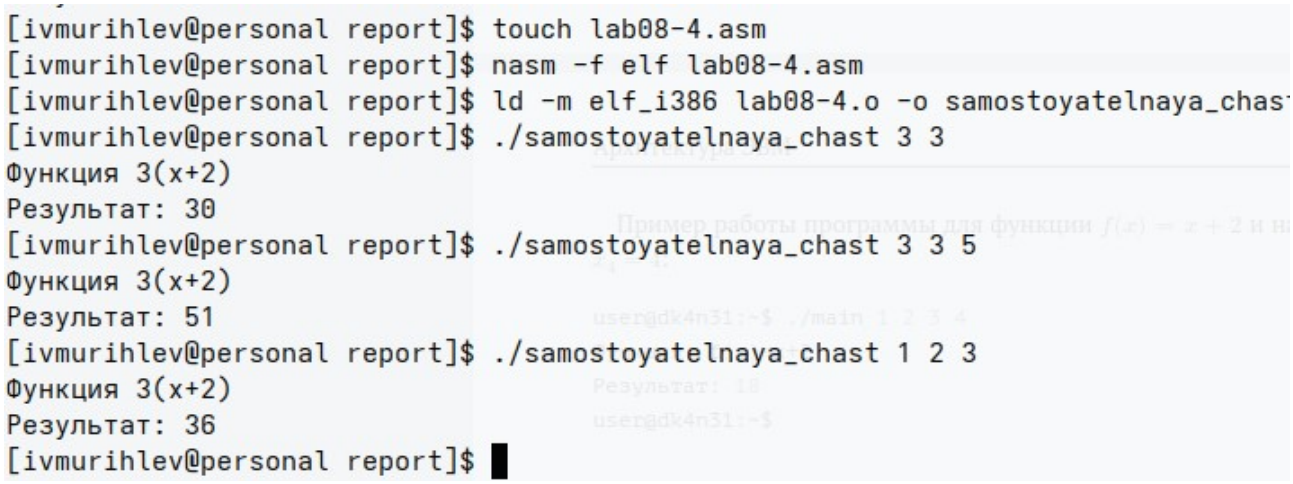
Самостоятельная часть

Создаю файл lab08-4.asm и пишу программу для 7 варианта,то есть f(x) =3(x+2). Затем делую из него исполняемый файлю(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:

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

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

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



Вывод

Была изучмна реализация механизма циклов на языке Assambler и структура стека в данном языке и была написанна программа с использованием этого механизма.(я не знаю что писать толком)