

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: Метвалли Ахмед Фарг Набеев

Группа: НПИбд-02-23

МОСКВА 2023г

Содержание

1.Цель работы

2.Задание

3.Теоретическое введение

4.Выполнение самостоятельной работы

5.Выводы

Список иллюстраций

3.1 Создание каталога и переход в него

Ошибка! Закладка не определена.

3.2 Создание файла

3.3 Ввод программы

3.4 Запуск файла

3.5 Изменение программы

3.6 Изменение программы

3.7 Запуск файла

3.8 Изменение программы

3.9 Запуск файла

3.10 Создание файла

3.11 Ввод программы

3.12 Запуск файла

3.13 Создание файла

3.14 Ввод программы

3.15 Запуск файла

3.16 Изменение программы

3.17 Запуск файла

4.1 Создание файла

4.2 Ввод программы

4.3 Проверка

Список таблиц

1 . Цель работы

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

2 . Задание

Здесь приводится описание задания в соответствии с рекомендациями методического пособия и выданным вариантом.

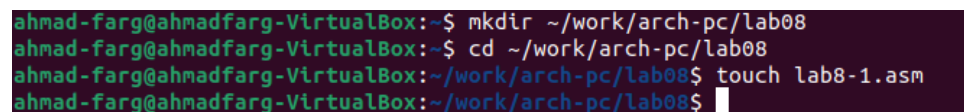
3.Теоретическое введение

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. На рис. 8.1 показана схема организации стека в процессоре. 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). # Выполнение лабораторной работы создал каталог work/arch-pc/lab08 и перешел в него (рис. 3.1).

Рис. 3.1: Создание каталога и переход в него

Создал файл в каталоге (рис. 3.2).



```
ahmad-farg@ahmadfarg-VirtualBox:~$ mkdir ~/work/arch-pc/lab08
ahmad-farg@ahmadfarg-VirtualBox:~$ cd ~/work/arch-pc/lab08
ahmad-farg@ahmadfarg-VirtualBox:~/work/arch-pc/lab08$ touch lab8-1.asm
ahmad-farg@ahmadfarg-VirtualBox:~/work/arch-pc/lab08$
```

Рис. 3.2: Создание файла

Ввел программу в файл lab8-1.asm (рис. 3.3).

```

GNU nano 6.2
#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. 3.3: Ввод программы

Запустил исполняемый файл (рис. 3.4).

```

ahmad-farg@ahmadfarg-VirtualBox:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 10
9
8
7
6
5
4
3
2
1
0

```



```

Введите N: 3
3
2
1

```

Рис. 3.4: Запуск файла

Изменил текст программы добавив изменение значение регистра ecx в цикле (рис. 3.5).

Рис. 3.5: Изменение программы

Редактировал программу в файле lab8-1.asm (рис. 3.6).

```

:~$ mkdir ~/work/arch-pc/lab08
:~$ cd ~/work/arch-pc/lab08
:~/work/arch-pc/lab08$ touch lab8-1.asm
:~/work/arch-pc/lab08$

```

Рис. 3.6: Изменение программы

Запустил исполняемый файл и получил такой результат(рис. 3.7).

```

GNU nano 6.2 /hc
#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. 3.7: Запуск файла

Внес изменения в текст программы добавив команды push и pop (рис. 3.8)

Рис. 3.8: Изменение программы

Запустил исполняемый файл (рис. 3.9)

Рис. 3.9: Запуск файла

Создал файл lab8-2.asm в том же каталоге (рис. 3.10).

```
ahmad-farg@ahmadfarg-VirtualBox:~/work/arch-pc/lab08$ touch lab8-2.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, 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 eax, esi ; записываем сумму в регистр `eax`
call iprintLF ; печать результата
call quit ; завершение программы
```

```
ahmad-farg@ahmadfarg-VirtualBox:~/work/arch-pc/lab08$ ./lab8-2 1 2 '3'
```

```
1
2
3
```

```

GNU nano 6.2
#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

```

Введите N: 3
3
2
1

```

#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 sprintLF ; вызываем функцию печати
loop next ; переход к обработке следующего
; аргумента (переход на метку 'next')
_end:
call quit

```

Рис. 3.10: Создание файла

Ввел программу в файл (рис. 3.11)

Рис. 3.11: Ввод программы

Запустил исполняемый файл указав его аргументы (рис. 3.12).

Рис. 3.12: Запуск файла

Создал файл lab8-3.asm в том же каталоге (рис. 3.13).

Рис. 3.13: Создание файла

Ввел программу в файл lab8-3.asm (рис. 3.14).

Рис. 3.14: Ввод программы

Запустил исполняемый файл указав аргументы (рис. 3.15).

Рис. 3.15: Запуск файла

Изменил файл lab8-3.asm.

```
ahmad-farg@ahmadfarg-VirtualBox:~/work/arch-pc/lab08$ touch lab8-3.asm
```

```
ahmad-farg@ahmadfarg-VirtualBox:~/work/arch-pc/lab08$ ./lab8-3 5 5 4
Результат: 0
```

Рис. 3.16: Изменение программы

Запустил исполняемый файл.

Рис. 3.17: Запуск файла

```

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
mul esi ; добавляем к промежуточной сумме
mov esi,eax;
loop next ; переход к обработке следующего аргумента

```

4.Выполнение самостоятельной работы

Создал файл lab8-4.asm для выполнения самостоятельной работы (рис. 4.1).

```

ahmad-farg@ahmadfarg-VirtualBox:~/work/arch-pc/lab08$ touch lab8-4.asm

```

Рис. 4.1: Создание файла

Написал программу для нахождения суммы значений функции $2(x-1)$ (вариант 4) (рис. 4.2).

```

%include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
SECTION .bss
ans:RESB 80
SECTION .text
global _start
_start:
pop ecx
pop edx
sub ecx,1
mov esi, 10
next:
cmp ecx,0h
jz _end
pop eax
call atoi
mul esi
sub eax,5
add [ans],eax
loop next
_end:
mov eax, msg
call sprint
mov eax,[ans]
call iprintLF
call quit

```

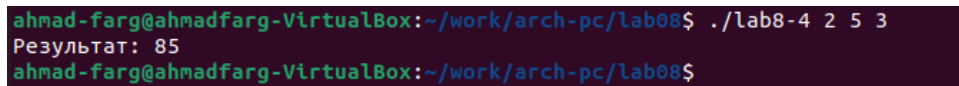
```

ahmad-farg@ahmadfarg-VirtualBox:~/work/arch-pc/lab08$ ./lab8-4 3 1 4
Результат: 65

```

Рис. 4.2: Ввод программы

Запускаю исполняемый файл с аргументами 1 2 3 4 (рис. 4.3).

A terminal window with a dark background. The prompt is 'ahmad-farg@ahmadfarg-VirtualBox:~/work/arch-pc/lab08\$'. The command './lab8-4 2 5 3' is entered. The output 'Результат: 85' is displayed. The prompt is repeated on the next line.

```
ahmad-farg@ahmadfarg-VirtualBox:~/work/arch-pc/lab08$ ./lab8-4 2 5 3
Результат: 85
ahmad-farg@ahmadfarg-VirtualBox:~/work/arch-pc/lab08$
```

Рис. 4.3: Проверка

5. Выводы

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