

Отчет по лабораторной работе №8

Дисциплина: архитектура компьютера

Авдеенко Марьяна Дмитриевна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
5	Задания самостоятельной работы	14
6	Выводы	16
	Список литературы	17

Список иллюстраций

4.1	Файл с лимтингом 8.1	8
4.2	Проверка файла	9
4.3	Файл с листингом 8.1 с изменениями	10
4.4	Проверка файла	10
4.5	Файл с листингом 8.2	11
4.6	Проверка файла	11
4.7	Файл с листингом 8.3	12
4.8	Проверка файла	12
4.9	Файл с листингом 8.3 с изменениями	13
4.10	Проверка файла	13
5.1	Файл с кодом	14
5.2	Проверка файла	15

Список таблиц

1 Цель работы

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

2 Задание

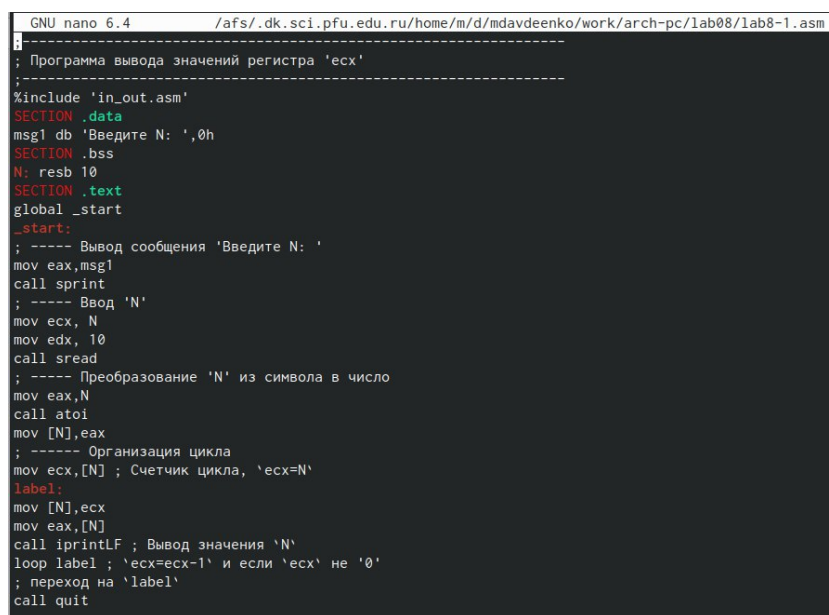
Здесь приводится описание задания в соответствии с рекомендациями методического пособия и выданным вариантом.

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: 1) добавление элемента в вершину стека (push); 2) извлечение элемента из вершины стека (pop).

4 Выполнение лабораторной работы

- 1) Открыла терминал.
- 2) Перешла в каталог, созданный для лабораторной работы №8 и создала lab8-1.asm.
- 3) Ввела в файл lab8-1.asm текст программы из листинга 8.1 (рис. 4.1).



```
GNU nano 6.4 /afs/.dk.sci.pfu.edu.ru/home/m/d/mdavdeenko/work/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. 4.1: Файл с листингом 8.1

- 4) Создала исполняемый файл и запустила его (рис. 4.2).


```

mdavdeenko@dk3n65 ~ $ cd work/arch-pc
mdavdeenko@dk3n65 ~/work/arch-pc $ mkdir lab08
mdavdeenko@dk3n65 ~/work/arch-pc $ cd lab08
mdavdeenko@dk3n65 ~/work/arch-pc/lab08 $ touch lab8-1.asm
mdavdeenko@dk3n65 ~/work/arch-pc/lab08 $ mc

mdavdeenko@dk3n65 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
lab8-1.asm:4: error: unable to open include file 'in_out.asm': No such file or directory
mdavdeenko@dk3n65 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
mdavdeenko@dk3n65 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
mdavdeenko@dk3n65 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 1
1
mdavdeenko@dk3n65 ~/work/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
mdavdeenko@dk3n65 ~/work/arch-pc/lab08 $ 

```

Рис. 4.2: Проверка файла

- 5) Внесла изменения в текст программы добавив команды `push` и `pop` (добавления в стек и извлечения из стека) для сохранения значения счетчика цикла `loop` (рис. 4.3).

```

GNU nano 6.4 /afs/.dk.sci.pfu.edu.ru/home/m/d/mdavdeenko/work/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:
push ecx ; добавление значения ecx в стек
sub ecx,1
mov [N],ecx
mov eax,[N]
call iprintLF
pop ecx ; извлечение значения ecx из стека
loop label

```

Рис. 4.3: Файл с листингом 8.1 с изменениями

- 6) Создала исполняемый файл и запустила его, используя 3 аргумента (рис. 4.4).

```

mdavdeenko@dk3n65 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
mdavdeenko@dk3n65 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
mdavdeenko@dk3n65 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 2
1
0

```

Рис. 4.4: Проверка файла

- 7) Создала файл lab8-2.asm в каталоге ~/work/arch-pc/lab08. Внимательно изучила текст программы из листинга 8.2 и ввела в lab8-2.asm. (рис. 4.5).

```

GNU nano 6.4 /afs/.dk.sci.pfu.edu.ru/home/m/d/mdavdeenko/work/arch-pc/lab08/lab8-2.asm
;-----
; 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. 4.5: Файл с листингом 8.2

8) Создала исполняемый файл и проверила его работу (рис. 4.6).

```

mdavdeenko@dk3n65 ~/work/arch-pc/lab08 $ nasm -f elf lab8-2.asm
mdavdeenko@dk3n65 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-2 lab8-2.o
mdavdeenko@dk3n65 ~/work/arch-pc/lab08 $ ./lab8-2 2 3 1
2
3
1
mdavdeenko@dk3n65 ~/work/arch-pc/lab08 $

```

Рис. 4.6: Проверка файла

9) Создала файл lab8-3.asm в каталоге ~/work/arch-pc/lab08. Внимательно изучила текст программы из листинга 8.3 и ввела в lab8-3.asm. (рис. 4.7).

```

GNU nano 6.4 /afs/.dk.sci.pfu.edu.ru/home/m/d/mdavdeenko/work/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, 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 iprintf ; печать результата
call quit ; завершение программы

```

Рис. 4.7: Файл с листингом 8.3

10) Создала исполняемый файл и проверила его работу (рис. 4.8).

```

mdavdeenko@dk3n65 ~/work/arch-pc/lab08 $ nasm -f elf lab8-3.asm
mdavdeenko@dk3n65 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-3 lab8-3.o
mdavdeenko@dk3n65 ~/work/arch-pc/lab08 $ ./lab8-3 12 13 7 10 5
Результат: 47

```

Рис. 4.8: Проверка файла

11) Внесла изменения в текст программы, изменив функцию сложения на умножение (рис. 4.9).

```

/afs/.dk.sci.pfu.edu.ru/home/m/d/mdavdeenko/work/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 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
mul esi ; добавляем к промежуточной сумме
; след. аргумент 'esi=esi+eax'
mov 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

```

Рис. 4.9: Файл с листингом 8.3 с изменениями

12) Создала исполняемый файл и проверила его работу (рис. 4.10).

```

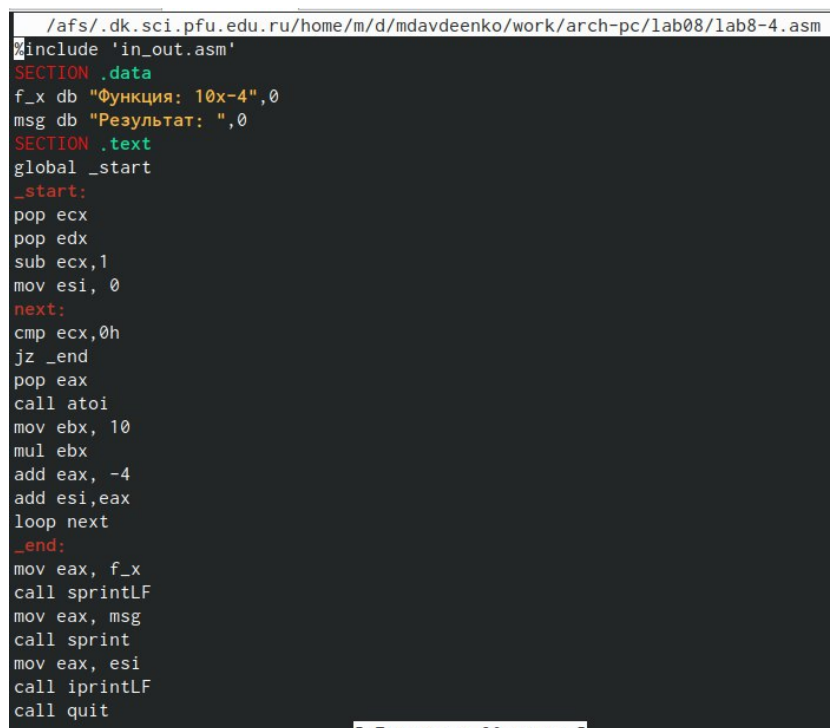
mdavdeenko@dk3n37 ~/work/arch-pc/lab08 $ nasm -f elf lab8-3.asm
mdavdeenko@dk3n37 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-3 lab8-3.o
mdavdeenko@dk3n37 ~/work/arch-pc/lab08 $ ./lab8-3 1 2 3
Результат: 6

```

Рис. 4.10: Проверка файла

5 Задания самостоятельной работы

- 9) Напишите программу, которая находит сумму значений функции $f(x)$ для $x = x_1, x_2, \dots, x_n$, т.е. программа должна выводить значение $f(x_1) + f(x_2) + \dots + f(x_n)$. Значения x_i передаются как аргументы. Вид функции $f(x)$ выбрать из таблицы 8.1 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 7 (рис. 5.1).



```
/afs/.dk.sci.pfu.edu.ru/home/m/d/mdavdeenko/work/arch-pc/lab08/lab8-4.asm
%include 'in_out.asm'
SECTION .data
f_x db "Функция: 10x-4",0
msg db "Результат: ",0
SECTION .text
global _start
_start:
pop ecx
pop edx
sub ecx,1
mov esi, 0
next:
cmp ecx,0h
jz _end
pop eax
call atoi
mov ebx, 10
mul ebx
add eax, -4
add esi,eax
loop next
_end:
mov eax, f_x
call sprintf
mov eax, msg
call sprintf
mov eax, esi
call iprintLF
call quit
```

Рис. 5.1: Файл с кодом

- 10) Создала исполняемый файл и проверила его работу на нескольких наборах $x = x_1, x_2, \dots, x_n$ (рис. 5.2).

```
mdavdeenko@dk3n37 ~/work/arch-pc/lab08 $ nasm -f elf lab8-4.asm
mdavdeenko@dk3n37 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-4 lab8-4.o
mdavdeenko@dk3n37 ~/work/arch-pc/lab08 $ ./lab8-4 1 2 2
Функция: 10x-4
Результат: 38
```

Рис. 5.2: Проверка файла

6 Выводы

В ходе данной лабораторной работы были приобретены навыки написания программ с использованием циклов и обработкой аргументов командной строки.

Список литературы