

Лабораторная работа №8

Архитектура компьютеров

Тойчубекова Асель Нурлановна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
3.1	Организация стека	7
3.2	Добавление элемента в стек	7
3.3	Извлечение элемента из стека	7
3.4	Инструкции организации циклов	8
4	Выполнение лабораторной работы	9
4.1	Задание для самостоятельной работы	17
5	Выводы	21
	Список литературы	22

Список иллюстраций

4.1	РИС.1 Создание каталога и файла	9
4.2	РИС.2 Редактирование файла	10
4.3	РИС.3 Запуск исполняемого файла	10
4.4	РИС.4 Редактирование файла	11
4.5	РИС.5 Запуск исполняемого файла	12
4.6	РИС.5_1 Запуск исполняемого файла	12
4.7	РИС.6 Редактирование файла	13
4.8	РИС.7 Запуск исполняемого файла	13
4.9	РИС.8 Создание файла	14
4.10	РИС.9 Редактирование файла	14
4.11	РИС.10 Запуск исполняемого файла	14
4.12	РИС.11 Создание файла	15
4.13	РИС.11_1 Редактирование файла	15
4.14	РИС.12 Запуск исполняемого файла	15
4.15	РИС.13 Редактирование файла	17
4.16	РИС.14 Запуск исполняемого файла	17
4.17	РИС.15 Редактирование файла	19
4.18	РИС.16 Запуск исполняемого файла	20

Список таблиц

1 Цель работы

Целью данной лабораторной работы является приобретение навыков написания программ с использованием циклов и обработкой аргументов командной строки.

2 Задание

1. Изучить теоретическое введение:

- Организация стека;
- Добавление элемента в стек;
- Извлечение элемента из стека;
- Инструкции организации циклов.

2. Изучив реализацию циклов в NASM, написать программу вывода значений регистра ехх.

3. Изучив обработку аргументов командной строки, написать программу выводящую на экран аргументы командной строки.

4. Написать программу вычисляющую суммы и произведений аргументов командной строки.

5. Выполнить задание для самостоятельной работы:

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 No 7. Создайте исполняемый файл и проверьте его работу на нескольких наборах $x = x_1, x_2, \dots, x_n$

3 Теоретическое введение

3.1 Организация стека

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. Для стека существует две основные операции: `push`- добавление элементов в вершину стека; `pop`-извлечение элемента из вершины стека.

3.2 Добавление элемента в стек

Команда `push` размещает значение в стеке, т.е. помещает значение в ячейку памяти, на которую указывает регистр `esp`, после этого значение регистра `esp` увеличивается на 4. Данная команда имеет один операнд — значение, которое необходимо поместить в стек.

3.3 Извлечение элемента из стека

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. Нужно помнить, что извлечённый из стека элемент

не стирается из памяти и остаётся как “мусор”, который будет перезаписан при записи нового значения в стек

3.4 Инструкции организации циклов

Для организации циклов существуют специальные инструкции. Для всех инструкций максимальное количество проходов задаётся в регистре есх. Наиболее простой является инструкция loop. Она позволяет организовать безусловный цикл, типичная структура. Инструкция loop выполняется в два этапа. Сначала из регистра есх вычитается единица и его значение сравнивается с нулём. Если регистр не равен нулю, то выполняется переход к указанной метке. Иначе переход не выполняется и управление передаётся команде, которая следует сразу после команды loop.

4 Выполнение лабораторной работы

Для начала создадим каталог для программ лабораторной работы №8. Перейдя в каталог создадим файл lab8-1.asm.(РИС.1)

```
antoychubekova@dk5n59 ~ $ mkdir ~/work/arch-pc/lab08
antoychubekova@dk5n59 ~ $ cd ~/work/arch-pc/lab08
antoychubekova@dk5n59 ~/work/arch-pc/lab08 $ touch lab8-1.asm
antoychubekova@dk5n59 ~/work/arch-pc/lab08 $ ls
lab8-1.asm
antoychubekova@dk5n59 ~/work/arch-pc/lab08 $
```

Рис. 4.1: РИС.1 Создание каталога и файла

Напишем в файл lab8-1.asm программу вывода значений регистра eax, перед этим внимательно изучив ее.(РИС.2)

```

lab8-1.asm      [-M--] 20 L:[ 1+26 27/ 28] *(626 / 636b) 0010 0x00A
%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. 4.2: РИС.2 Редактирование файла

Создадим исполняемый файл и запустим его. Мы видим, что программа выводит значения от N до 1.(РИС.3) Данный пример показывает, что использование регистра ecx в теле цикла loop может привести к некорректной работе программы

```

antoychubekova@dk6n55 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
antoychubekova@dk6n55 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
antoychubekova@dk6n55 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 6
6
5
4
3
2
1

```

Рис. 4.3: РИС.3 Запуск исполняемого файла

Чтобы исправить некорректную работу программы внесем изменения в текст программы добавив изменение значение регистра ecx в цикле.(РИС.4)

```

lab8-1.asm      [-M--] 36 L:[ 5+16 21/ 28] *(465 / 636b) 0010 0x00A
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:
sub [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. 4.4: РИС.4 Редактирование файла

Создадим исполняемый файл и запустим его. Мы видим, что программа запускает бесконечный цикл при нечетном значении N (РИС.5) и выводит только нечетные числа при четном значении N.(РИС.5_1)

```

4294295848
4294295846
4294295844
4294295842
4294295840
4294295838
4294295836
4294295834
4294295832
4294295830
4294295828
4294295826
4294295824
42942958^Z
[1]+  Остановлен      ./lab8-1
antoyjchubekova@dk8n54 ~/work/arch-pc/lab08 $

```

Рис. 4.5: РИС.5 Запуск исполняемого файла

```

antoyjchubekova@dk6n55 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
antoyjchubekova@dk6n55 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
antoyjchubekova@dk6n55 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 6
5
3
1
Ошибка сегментирования (стек памяти сброшен на диск)

```

Рис. 4.6: РИС.5_1 Запуск исполняемого файла

Для корректной работы программы внесем изменения в текст программы добавив push и pop для сохранения значения счетчика цикла loop.(РИС.6)

```

lab8-1.asm      [-M--]  9 L:[ 1+29 30/ 30] *(674 / 674b) <EOF>
%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
call quit

```

Рис. 4.7: РИС.6 Редактирование файла

Создадим исполняемый файл и запустим его. Мы видим, что программа выводит числа от N-1 до 1, где количество проходов цикла соответствует значению N.(РИС.7)

```

antoyjchubekova@dk8n54 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
antoyjchubekova@dk8n54 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
antoyjchubekova@dk8n54 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 7
6
5
4
3
2
1
0

```

Рис. 4.8: РИС.7 Запуск исполняемого файла

Создадим файл lab8-2.asm.(РИС.8) Внимательно изучив, запишем в этот файл программу выводящую на экран аргументы командной строки.(РИС.9)

```
antoyjchubekova@dk8n54 ~/work/arch-pc/lab08 $ touch lab8-2.asm
```

Рис. 4.9: РИС.8 Создание файла

```
lab8-2.asm      [-M--]  9 L:[ 1+19 20/ 20] *(943 / 943b) <EOF>
%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. 4.10: РИС.9 Редактирование файла

Создадим исполняемый файл и запустим его. Программа обработала 4 аргумента, так как аргумент 2 не был взят в кавычки, в отличие от аргумента 3, поэтому из-за пробела программа берет 2 как отдельный аргумент.(РИС.10)

```
antoyjchubekova@dk8n54 ~/work/arch-pc/lab08 $ nasm -f elf lab8-2.asm
antoyjchubekova@dk8n54 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-2 lab8-2.o
antoyjchubekova@dk8n54 ~/work/arch-pc/lab08 $ ./lab8-2 аргумент1 аргумент 2 'аргумент 3
аргумент1
аргумент
2
аргумент 3
antoyjchubekova@dk8n54 ~/work/arch-pc/lab08 $
```

Рис. 4.11: РИС.10 Запуск исполняемого файла

Создадим файл lab8-3.asm.(РИС.11) Внимательно изучив, запишем в него программу вычисления суммы аргументов командной строки.(РИС.11)

```

antoychubekova@dk8n54 ~/work/arch-pc/lab08 $ touch lab8-3.asm
antoychubekova@dk8n54 ~/work/arch-pc/lab08 $ ls
in_out.asm lab8-1 lab8-1.asm lab8-1.o lab8-2 lab8-2.asm lab8-2.o lab8-3.asm
antoychubekova@dk8n54 ~/work/arch-pc/lab08 $

```

Рис. 4.12: РИС.11 Создание файла

```

lab8-3.asm [-M--] 50 L:[ 1+ 8 9/ 29] *(327 /1428b) 0010 0x00A
#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

```

Рис. 4.13: РИС.11_1 Редактирование файла

Создадим исполняемый файл и запустим его. Проверим работу программы на значениях (12 13 7 10 5) и на значениях (7 4 6 9 10). Мы видим, что все работает правильно. (РИС.12)

```

antoychubekova@dk8n54 ~/work/arch-pc/lab08 $ nasm -f elf lab8-3.asm
antoychubekova@dk8n54 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-3 lab8-3.o
antoychubekova@dk8n54 ~/work/arch-pc/lab08 $ ./lab8-3 12 13 7 10 5
Результат: 47
antoychubekova@dk8n54 ~/work/arch-pc/lab08 $ nasm -f elf lab8-3.asm
antoychubekova@dk8n54 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-3 lab8-3.o
antoychubekova@dk8n54 ~/work/arch-pc/lab08 $ ./lab8-3 7 4 6 9 10
Результат: 36

```

Рис. 4.14: РИС.12 Запуск исполняемого файла

Изменим текст программы так, чтобы программа вычисляла произведения аргументов командной строки(РИС.13) Сама программа выглядит следующим образом:

```
%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
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.

```
lab8-3.asm          [-M--] 32 L:[ 1+28 29/ 29] *(1332/1332b) <EOF>
#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
mov esi,eax
loop next ; переход к обработке следующего аргумента
_end:
mov eax, msg ; вывод сообщения "Результат: "
call sprintf
mov eax, esi ; записываем сумму в регистр 'eax'
call iprintLF ; печать результата
call quit ; завершение программы
```

Рис. 4.15: РИС.13 Редактирование файла

Создадим исполняемый файл и запустим его. Проверим работу программы на значениях (1 2 3 4 5). Мы видим, что все правильно работает(РИС.14)

```
antoychubekova@dk8n54 ~/work/arch-pc/lab08 $ nasm -f elf lab8-3.asm
antoychubekova@dk8n54 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-3 lab8-3.o
antoychubekova@dk8n54 ~/work/arch-pc/lab08 $ ./lab8-3 1 2 3 4 5
Результат: 120
```

Рис. 4.16: РИС.14 Запуск исполняемого файла

4.1 Задание для самостоятельной работы

Напишем программу которая находит сумму значений функции $f(x)=7(x+1)$ (вариант 14) для $x=x_1, x_2, \dots, x_n$, то есть программа должна выводить $F(x_1) + f(x_2) + \dots +$

f(xn)(РИС.15) Сама программа выглядит следующим образом:

```
%include 'in_out.asm'
```

```
SECTION .data  
msg db "Результат:",0
```

```
SECTION .text  
global _start  
_start:
```

```
    pop ecx
```

```
    pop edx
```

```
    sub ecx,1  
    mov esi,0
```

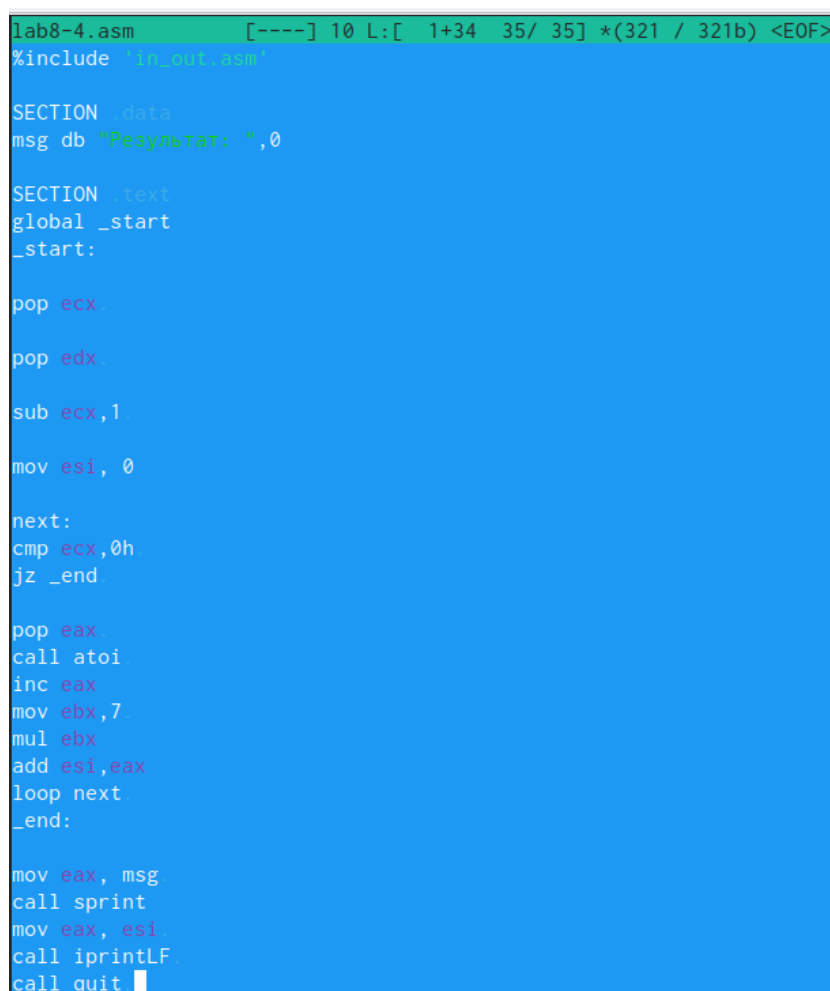
```
    next:  
    cmp ecx,0h  
    jz _end
```

```
    pop eax  
    call atoi inc eax  
    mov ebx,7  
    mul ebx  
    add esi,eax  
    loop next  
_end:
```

```

mov eax, msg
call sprint
mov eax, esi call iprintLF call quit

```



```

lab8-4.asm [----] 10 L: [ 1+34 35/ 35] *(321 / 321b) <EOF>
#include 'in_out.asm'

SECTION .data
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
inc eax
mov ebx,7
mul ebx
add esi,eax
loop next
_end:

mov eax, msg
call sprint
mov eax, esi
call iprintLF
call quit

```

Рис. 4.17: РИС.15 Редактирование файла

Создадим исполняемый файл и запустим его. Проверим корректность работы программы введя значения (1 2 3);(6 7 8);(4 3). Используя калькулятор и проверив правильный ответ мы видим, что программа работает правильно.(РИС.16)

```
antoyjchubekova@dk5n58 ~/work/arch-pc/lab08 $ nasm -f elf lab8-4.asm
antoyjchubekova@dk5n58 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-4 lab8-4.o
antoyjchubekova@dk5n58 ~/work/arch-pc/lab08 $ ./lab8-4 1 2 3
Результат: 63
antoyjchubekova@dk5n58 ~/work/arch-pc/lab08 $ ./lab8-4 6 7 8
Результат: 168
antoyjchubekova@dk5n58 ~/work/arch-pc/lab08 $ ./lab8-4 4 3
Результат: 63
antoyjchubekova@dk5n58 ~/work/arch-pc/lab08 $
```

Рис. 4.18: РИС.16 Запуск исполняемого файла

5 Выводы

В ходе выполнения лабораторной работы №8 я приобрела навыки написания программ с использованием циклов и обработкой аргументов командной строки. Применяя полученные навыки написала программу, которая находит значение сумму значений функции $f(x)$ для различных x .

Список литературы

-<https://esystem.rudn.ru/course/view.php?id=4975>.