

Шаблон отчёта по лабораторной работе N8

Дициплина:Архитектура компьютеров

Аннаоразов Сердар Аннаоразович

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	9
4.1	Обработка аргументов командной строки	13
5	Задание для самостоятельной работы	16
6	Выводы	18
	Список литературы	19

Список иллюстраций

4.1	Создания каталога и файла	9
4.2	Копирования файла in_out.asm	9
4.3	Введения кода в файл	10
4.4	Создания выполняемого файла lab8-1.asm	10
4.5	Создания копии файла lab8-1.asm	11
4.6	Исправления текста в файле	11
4.7	Создания выполняемого файла lab8-1-1.asm	12
4.8	Создания второй копии файла lab8-1.asm	12
4.9	Исправления текста в файле lab8-1-2.asm	13
4.10	Создания выполняемого файла lab8-1-2.asm	13
4.11	Создания файла	13
4.12	Программа выводющая на экран аргументы командной строки . .	14
4.13	Создания исполняемого файла и запуск его с указанными аргументами	14
4.14	Скопирования файла lab8-2-1	15
4.15	программа которая выводит сумму всех решений примера	15
5.1	Создания файла для самостоятельной работы	16
5.2	программа которая выводит сумму всех решений примера	17
5.3	Результаты работы	17

Список таблиц

1 Цель работы

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

2 Задание

1. Реализация циклов в NASM
2. Обработка аргументов командной строки

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).

Для организации циклов существуют специальные инструкции. Для всех инструкций максимальное количество проходов задаётся в регистре ecx. Наиболее простой является инструкция loop. Она позволяет организовать безусловный цикл, типичная структура которого имеет следующий вид: mov ecx, 100 ; Количество проходов NextStep: ... ; тело цикла ... loop NextStep ; Повторить ecx раз от метки NextStep Инструкция loop выполняется в два этапа. Сначала из регистра ecx 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

##Реализация циклов в NASM Для начала я создал каталог для программам лабораторной работы № 8, перешёл в него и создал файл lab8-1.asm:

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

Рис. 4.1: Создания каталога и файла

После этого как я делал в предыдущих лабораторных работ я скопирую файл in_out.asm в новый созданный каталог

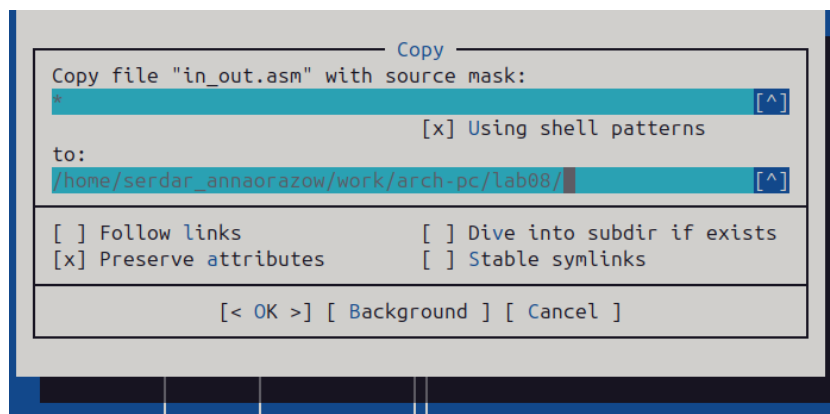


Рис. 4.2: Копирования файла in_out.asm

После этого я открыл новый созданный файл и написал туда код которая выводит значение регистра esx.

```
serdar_annaorazow@sc
GNU nano 7.2 /home/serdar_annaorazow
%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
;----- Вывод 'N'
mov ecx,N
mov edx, 10
call sread

; Преобразования 'N' из символа в число
mov eax,N
call atoi
mov [N],eax

;----- Организация цикла
mov ecx,[N]
label:
mov [N],ecx
mov eax,[N]
call iprintLF
loop label
call quit
```

Рис. 4.3: Введения кода в файл

Потом я создал выполняемый файл и запустил его

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

Рис. 4.4: Создания выполняемого файла lab8-1.asm

Потом я создал копию файла с именем lab8-1-1.asm

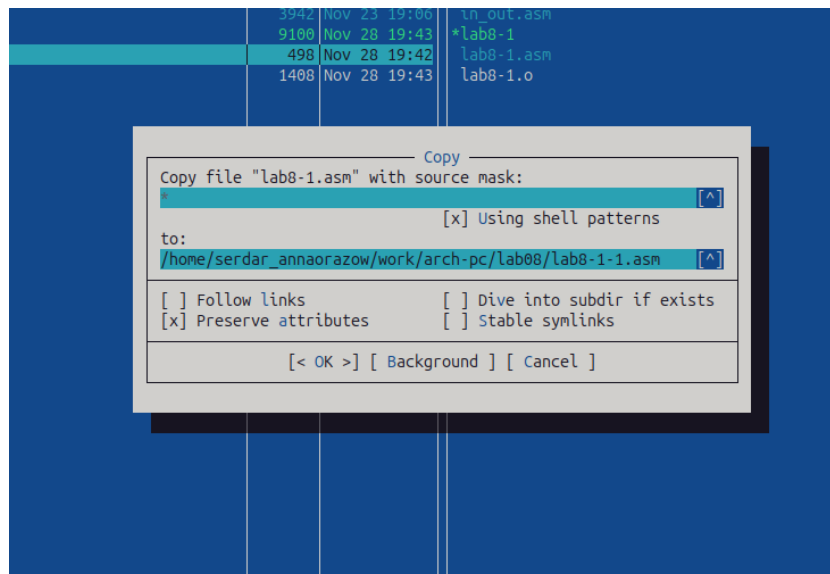


Рис. 4.5: Создания копии файла lab8-1.asm

Я открыл новый созданный файл и в нем изменил текст так использование регистра ecx в теле цикла loop может привести к некорректной работе программы

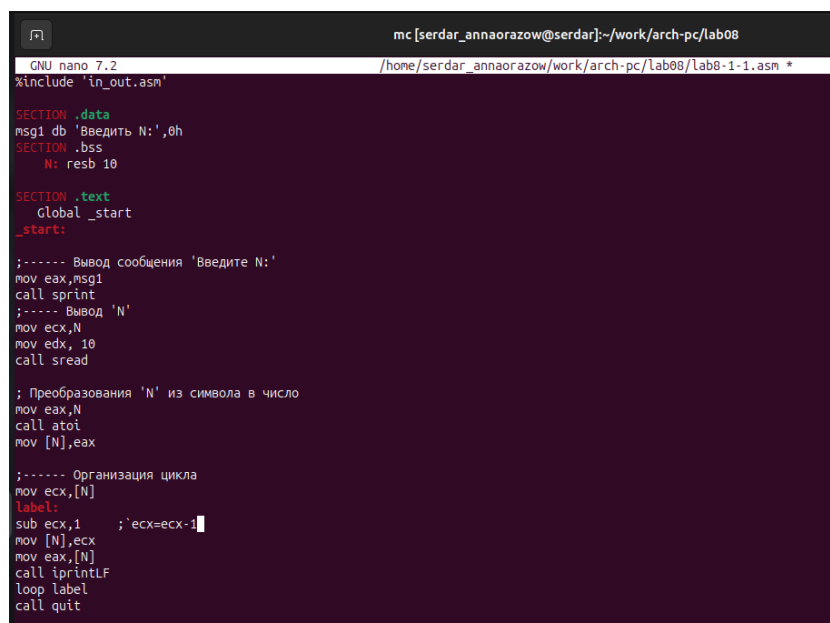


Рис. 4.6: Исправления текста в файле

Потом я создал выполняемый файл и запустил его

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

Рис. 4.7: Создания выполняемого файла lab8-1-1.asm

Потом я создал новую копию файла с именем lab8-1-2.asm

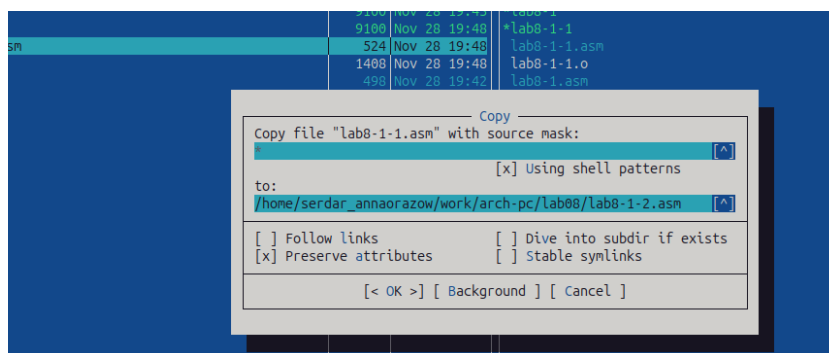


Рис. 4.8: Создания второй копии файла lab8-1.asm

После этого для использования регистра `ecx` в цикле и сохранения корректности работы программы использовал стек. Внес изменения в текст программы добавив команды `push` и `pop` (добавления в стек и извлечения из стека) для сохранения значения счетчика цикла `loop`:

```
serdar_annaorazow@serdar: ~/work/arch-pc/lab08
GNU nano 7.2 /home/serdar_annaorazow/work/arch-pc/lab08/lab8-1-2.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
;----- Вывод 'N'
mov ecx,N
mov edx,10
call sread

; Преобразования 'N' из символа в число
mov eax,N
call atoi
mov [N],eax

;----- Организация цикла
mov ecx,[N]
label:
push ecx
sub ecx,1
mov [N],ecx
mov eax,[N]
call iprintf
pop ecx
loop label
call quit
```

Рис. 4.9: Исправления текста в файле lab8-1-2.asm

Потом создал исполняемый файл и запустил его

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

Рис. 4.10: Создания выполняемого файла lab8-1-2.asm

4.1 Обработка аргументов командной строки

Для начало я создал новый файл для написание программы которая выводит на экран аргументы командной строки.

```
serdar_annaorazow@serdar:~/work/arch-pc/lab08$ touch lab8-2.asm
serdar_annaorazow@serdar:~/work/arch-pc/lab08$
```

Рис. 4.11: Создания файла

После этого я написал программу в созданный файл

```
GNU nano 7.2 serdar_annaorazow@serdar: ~/work/arch-pc/lab08
%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 printf ; вызываем функцию печати
    loop next ; переход к обработке следующего
             ; аргумента (переход на метку 'next')

_end:
    call quit
```

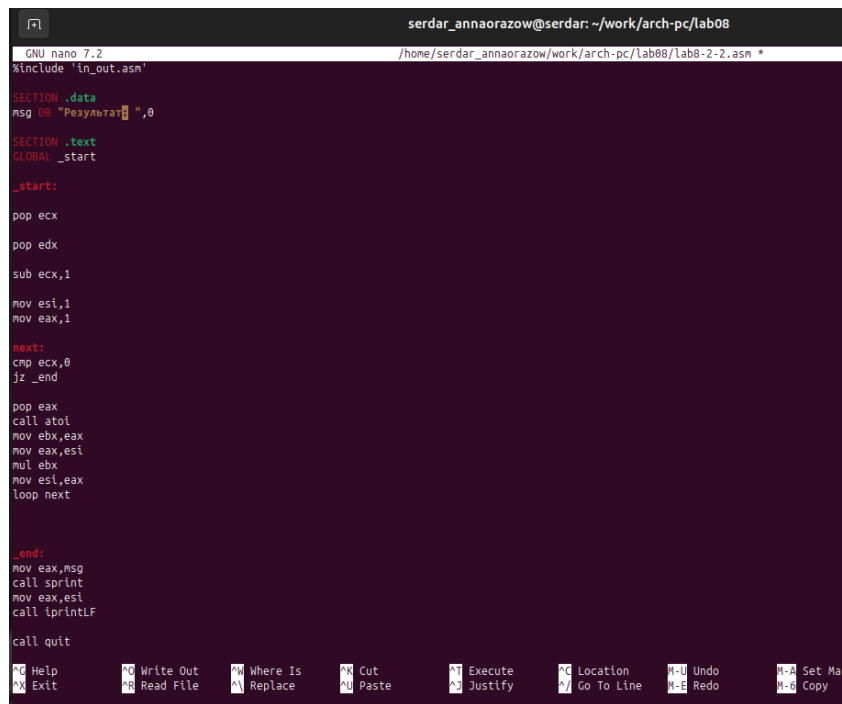
Рис. 4.12: Программа выводящая на экран аргументы командной строки

Создал исполняемый файл и запустил его, указав аргументы

```
serdar_annaorazow@serdar: ~/work/arch-pc/lab08$ nasm -f elf lab8-2.asm
serdar_annaorazow@serdar: ~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-2 lab8-2.o
serdar_annaorazow@serdar: ~/work/arch-pc/lab08$ ./lab8-2 аргумент1 аргумент2 'аргумент3'
аргумент1
аргумент2
аргумент3
serdar_annaorazow@serdar: ~/work/arch-pc/lab08$
```

Рис. 4.13: Создания исполняемого файла и запуск его с указанными аргументами

Потом я скопировал файл и изменил программу, чтобы она выводила произведение введенных чисел.



```
GNU nano 7.2 /home/serdar_annaorazow/work/arch-pc/lab08/lab8-2-2.asm *
#include 'in_out.asm'

SECTION .data
msg DB "Результат: ",0

SECTION .text
GLOBAL _start

_start:
    pop ecx
    pop edx
    sub ecx,1
    mov esi,1
    mov eax,1

next:
    cmp ecx,0
    jz _end

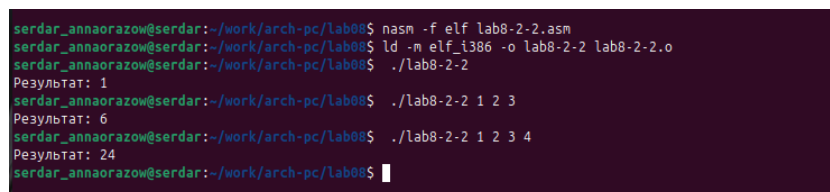
    pop eax
    call atoi
    mov ebx,eax
    mov eax,esi
    mul ebx
    mov esi,eax
    loop next

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

Help Exit Write Out Read File Where Is Replace Cut Paste Execute Justify Location Go To Line Undo Redo Set Ma Copy
```

Рис. 4.14: Скопирования файла lab8-2-1

После этого я создал исполняемый файл и запустил его.

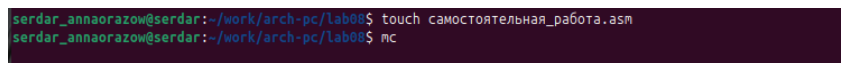


```
serdar_annaorazow@serdar:~/work/arch-pc/lab08$ nasm -f elf lab8-2-2.asm
serdar_annaorazow@serdar:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-2-2 lab8-2-2.o
serdar_annaorazow@serdar:~/work/arch-pc/lab08$ ./lab8-2-2
Результат: 1
serdar_annaorazow@serdar:~/work/arch-pc/lab08$ ./lab8-2-2 1 2 3
Результат: 6
serdar_annaorazow@serdar:~/work/arch-pc/lab08$ ./lab8-2-2 1 2 3 4
Результат: 24
serdar_annaorazow@serdar:~/work/arch-pc/lab08$
```

Рис. 4.15: программа которая выводит сумму всех решений примера

5 Задание для самостоятельной работы

Для начала я создал файл для самостоятельной работы



```
serdar_annaorazow@serdar:~/work/arch-pc/lab08$ touch самостоятельная_работа.asm
serdar_annaorazow@serdar:~/work/arch-pc/lab08$ mc
```

Рис. 5.1: Создания файла для самостоятельной работы

Я написал программу, которая выводит сумму всех решений примера. В лабораторной работе №7, я получил 10 вариантов, поэтому я писал программу для первого варианта. Введенные числа я придумал сам, и посчитал их, чтобы проверить работу программы.



```
serdar_annaorazow@serdar: ~/work/arch-pc/lab08
GNU nano 7.2 /home/serdar_annaorazow/work/arch-pc/lab08/самостоятельная_работа.
%include 'in_out.asm'

SECTION .data
prim DB 'f(x)=5(2+x)',0
otv DB 'Результат: ',0
SECTION .text
GLOBAL _start
_start:

pop ecx

pop edx

sub ecx,1

mov esi,0

mov eax,prim
call sprintfLF
next:
cmp ecx,0
jz _end

mov ebx,5
pop eax
call atoi
mul ebx

add eax,2

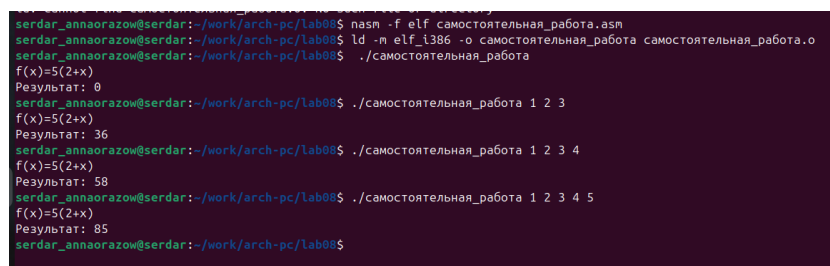
add esi,eax

loop next

_end:
mov eax,otv
call sprintf
mov eax,esi
call iprintfLF
call quit
```

Рис. 5.2: программа которая выводит сумму всех решений примера

Потом я создал исполняемый файл и запустил его, с аргументами. Проверил файл все ли правильно сделано



```
serdar_annaorazow@serdar: ~/work/arch-pc/lab08$ nasm -f elf самостоятельная_работа.asm
serdar_annaorazow@serdar: ~/work/arch-pc/lab08$ ld -m elf_i386 -o самостоятельная_работа.о самостоятельная_работа.o
serdar_annaorazow@serdar: ~/work/arch-pc/lab08$ ./самостоятельная_работа
f(x)=5(2+x)
Результат: 0
serdar_annaorazow@serdar: ~/work/arch-pc/lab08$ ./самостоятельная_работа 1 2 3
f(x)=5(2+x)
Результат: 36
serdar_annaorazow@serdar: ~/work/arch-pc/lab08$ ./самостоятельная_работа 1 2 3 4
f(x)=5(2+x)
Результат: 58
serdar_annaorazow@serdar: ~/work/arch-pc/lab08$ ./самостоятельная_работа 1 2 3 4 5
f(x)=5(2+x)
Результат: 85
serdar_annaorazow@serdar: ~/work/arch-pc/lab08$
```

Рис. 5.3: Результаты работы

6 Выводы

Я приобрел навыки написания программы с использованием цикла.

Список литературы

https://esystem.rudn.ru/pluginfile.php/2089548/mod_resource/content/0/%D0%9B%D0%B0%D0%9D%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%E2%84%968.%D0%9F%D1%80%D0%BE%D1%86%D0%B8%D0%BA%D0%BB%D0%B0.%D0%9E%D0%B1%D1%80%D0%B0%D0%B1%D0%B0%D1%80%D0%B3%D1%83%D0%BC%D0%B5%D0%BD%D1%82%D0%BE%D0%B2%D0%BA%D0%BE%D0%BC%D0%B0%D0%BD%D0%B4%D0%BD%D0%BE%D0%B9%D1%81%D1%82%D1%80%D0%BE%D0%BA%D0%B8..pdf :::