

Отчёт по лабораторной работе №8

Дисциплина: Архитектура компьютера

Мутале Чали

Содержание

1	Цель работы.....	1
2	Задание	1
3	Выполнение лабораторной работы	1
4	Выполнение самостоятельной работы	9
5	Выводы.....	11
6	Список литературы.....	11

1 Цель работы

Цель лабораторной 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 Выполнение лабораторной работы

1. Реализация циклов в NASM

Создаю каталог для программ лабораторной работы № 8, перехожу в него и создаю файл lab8-1.asm:

```
(cmutale@cmutale)-[~]  
$ mkdir -p ~/work/arch-pc/lab08  
  
(cmutale@cmutale)-[~]  
$ cd ~/work/arch-pc/lab08  
  
(cmutale@cmutale)-[~/work/arch-pc/lab08]  
$ touch lab8-1.asm
```

Рис 1

В файле lab8-1.asm вставляю код программы, которая показывает, как инструкция `loop` использует регистр `ecx` в качестве счетчика и на каждом шаге уменьшает его значение на единицу:

```
GNU nano 7.2 /home/cmutale/work/arch-pc/lab08/lab8-1.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
```

Рис 2

Создаю исполняемый файл:

```
(cmutale@cmutale)-[~/work/arch-pc/lab08]
$ nasm -f elf lab8-1.asm

(cmutale@cmutale)-[~/work/arch-pc/lab08]
$ ld -m elf_i386 -o lab8-1 lab8-1.o
```

Рис 3

При запуске, программа выводит значение регистра `ecx`:

```
(cmutale@cmutale)-[~/work/arch-pc/lab08]
$ ./lab8-1
Введите N: 8
8
7
6
5
4
3
2
1
```

Рис 4

Изменяю текст программы, чтобы она показала, что использование регистра `ecx` в теле цикла `loop` может привести к некорректной работе программы:

```
GNU nano 7.2 /home/cmutale/work/arch-pc/lab0
; — 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 ecx, 1 ; `ecx=ecx-1`
mov [N], ecx
mov eax, [N]
call iprintLF
loop label ; `ecx=ecx-1` и если `ecx` не '0'
; переход на `label`

call quit
```

Рис 5

Создаю исполняемый файл:

```

(cmutale@cmutale)-[~/work/arch-pc/lab08]
$ nasm -f elf lab8-1.asm

(cmutale@cmutale)-[~/work/arch-pc/lab08]
$ ld -m elf_i386 -o lab8-1 lab8-1.o

```

Рис 6

При запуске, программа выводит бесконечное значение, которое не соответствует значению N введенному с клавиатуры:

```

(cmutale@cmutale)-[~/work/arch-pc/lab08]
$ ./lab8-1
Введите N: 1

```

Рис 7

```

4293272770
4293272768
4293272766
4293272764
4293272762
4293272760
4293272758
4293272756
4293272754
4293272752
4293272750
4293272748
4293272746
4293272744
4293272742
4293272740
4293272738
4293272736
4293272734
4293272732
4293272730
4293272728
4293272726
4293272724
429327

```

Рис 8

Для использования регистра еsx в цикле и сохранения корректности работы программы можно использовать стек. Изменяю текст программы добавив команды push и pop:

```

29 label:
30 push ecx ; добавление значения ecx в стек
31 sub ecx,1
32 mov [N],ecx
33 mov eax,[N]
34 call iprintLF
35 pop ecx ; извлечение значения ecx из стека
36 loop label
37
38 call quit

```

Рис 9

Создаю исполняемый файл:

```

(cmutale@cmutale)-[~/work/arch-pc/lab08]
$ nasm -f elf lab8-1.asm

(cmutale@cmutale)-[~/work/arch-pc/lab08]
$ ld -m elf_i386 -o lab8-1 lab8-1.o

```

Рис 10

При запуске, программа выводит значение, которое соответствует значению N введенному с клавиатуры:

```

(cmutale@cmutale)-[~/work/arch-pc/lab08]
$ ./lab8-1
Введите N: 5

```

Рис 11

```

(cmutale@cmutale)-[~/work/arch-pc/lab08]
$ ./lab8-1
Введите N: 5
4
3
2
1
0

```

Рис 12

2. Обработка аргументов командной строки

Создаю файл lab8-2.asm:

```
(cmutale@cmutale)-[~/work/arch-pc/lab08] проверьте ех
$ touch lab8-2.asm по проходам цикла значению N введенному
(cmutale@cmutale)-[~/work/arch-pc/lab08]
$
```

Рис 13

Ввожу в него текст программы, которая выводит на экран аргументы командной строки:

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

SECTION .text
global _start
_start:
    ; Архитектура 386

    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
```

Рис 14

Создаю исполняемый файл и запускаю его:

```
(cmutale@cmutale)-[~/work/arch-pc/lab08]
$ nasm -f elf lab8-2.asm

(cmutale@cmutale)-[~/work/arch-pc/lab08]
$ ld -m elf_i386 -o lab8-2 lab8-2.o

(cmutale@cmutale)-[~/work/arch-pc/lab08]
$ ./lab8-2 5 6 '1'
5
6
1

(cmutale@cmutale)-[~/work/arch-pc/lab08]
$ ./lab8-2 5 b '1'
5
b
1
```

Рис 15

Создаю файл lab8-3.asm:

```
(cmutale@cmutale)-[~/work/arch-pc/lab08]
$ touch lab8-3.asm
```

Рис 16

Ввожу в него текст программы, которая выводит на экран сумму аргументов:

```
lab8-3.asm
~/work/arch-pc/lab08
Save

1 %include 'in_out.asm'
2
3 SECTION .data
4 msg db "Результат: ",0
5 SECTION .text
6 global _start
7 _start:
8
9 pop ecx ; Извлекаем из стека в `ecx` количество
10 ; аргументов (первое значение в стеке)
11 pop edx ; Извлекаем из стека в `edx` имя программы
12 ; (второе значение в стеке)
13 sub ecx,1 ; Уменьшаем `ecx` на 1 (количество
14 ; аргументов без названия программы)
15 mov esi, 0 ; Используем `esi` для хранения
16 ; промежуточных сумм
17
18 next:
19 cmp ecx,0h ; проверяем, есть ли еще аргументы
20 jz _end ; если аргументов нет выходим из цикла
21 ; (переход на метку `_end`)
22 pop eax ; иначе извлекаем следующий аргумент из стека
23 call atoi ; преобразуем символ в число
24 add esi,eax ; добавляем к промежуточной сумме
25 ; след. аргумент `esi=esi+eax`
26 loop next ; переход к обработке следующего аргумента
27
28 _end:
29 mov eax, msg
30 call sprint
31
32 mov eax, esi ; записываем сумму в регистр `eax`
33 call iprintLF
34
35 call quit
```

Рис 17

Создаю исполняемый файл и запускаю его:

```
(cmutale@cmutale)-[~/work/arch-pc/lab08]
$ nasm -f elf lab8-3.asm

(cmutilale@cmutale)-[~/work/arch-pc/lab08]
$ ld -m elf_i386 -o lab8-3 lab8-3.o
```

Рис 18

```
(cmutale@cmutale)-[~/work/arch-pc/lab08]
$ ./lab8-3 5 9 1
Результат: 15
```

Рис 19

Изменяю текст программы, чтобы она выводила произведение аргументов:

```
GNU nano 7.2
%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

    mov ebx, eax
    mov eax, esi
    mul ebx
    mov 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 20

Создаю исполняемый файл и запускаю его:

```
(cmutale@cmutale)-[~/work/arch-pc/lab08]
$ nasm -f elf lab8-3.asm

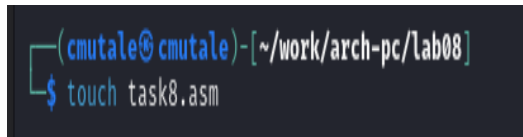
(cmutale@cmutale)-[~/work/arch-pc/lab08]
$ ld -m elf_i386 -o lab8-3 lab8-3.o

(cmutale@cmutale)-[~/work/arch-pc/lab08]
$ ./lab8-3 5 9
Результат: 45
```

Рис 21

4 Выполнение самостоятельной работы

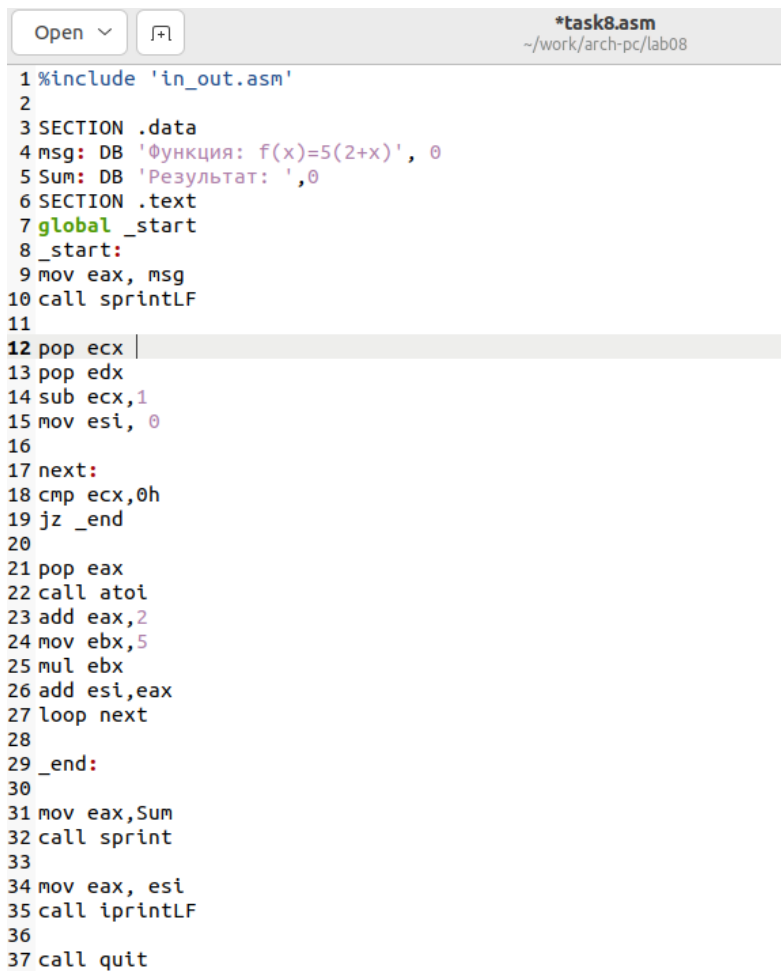
Создаю файл task8.asm:



```
(cmutale@cmutale)-[~/work/arch-pc/lab08]
$ touch task8.asm
```

Рис 22

В него пишу программу, которая находит сумму значений функции $f(x) = 5(2 + x)$ для некоторых значений x (вариант 10):



```
Open  [+]
```

***task8.asm**
~/work/arch-pc/lab08

```
1 %include 'in_out.asm'
2
3 SECTION .data
4 msg: DB 'Функция: f(x)=5(2+x)', 0
5 Sum: DB 'Результат: ', 0
6 SECTION .text
7 global _start
8 _start:
9 mov eax, msg
10 call sprintf
11
12 pop ecx
13 pop edx
14 sub ecx, 1
15 mov esi, 0
16
17 next:
18 cmp ecx, 0h
19 jz _end
20
21 pop eax
22 call atoi
23 add eax, 2
24 mov ebx, 5
25 mul ebx
26 add esi, eax
27 loop next
28
29 _end:
30
31 mov eax, Sum
32 call sprintf
33
34 mov eax, esi
35 call iprintLF
36
37 call quit
```

Рис 23

Код программы:

```
%include 'in_out.asm'

SECTION .data
msg: DB 'Функция: f(x)=5(2+x)', 0
```

```
Sum: DB 'Результат: ',0
```

```
SECTION .text
```

```
global _start
```

```
_start:
```

```
mov eax, msg
```

```
call sprintf
```

```
pop ecx
```

```
pop edx
```

```
sub ecx,1
```

```
mov esi, 0
```

```
next:
```

```
cmp ecx,0h
```

```
jz _end
```

```
pop eax
```

```
call atoi
```

```
add eax,2
```

```
mov ebx,5
```

```
mul ebx
```

```
add esi,eax
```

```
loop next
```

```
_end:
```

```
mov eax,Sum
```

```
call sprintf
```

```
mov eax, esi
```

```
call iprintLF
```

```
call quit
```

Создаю исполняемый файл и запускаю его. Программа выводит сумму $f(1)+f(2)+f(1)$:

```
(cmutale@cmutale)-[~/work/arch-pc/lab08]
$ nasm -f elf task8.asm

(cmutale@cmutale)-[~/work/arch-pc/lab08]
$ ld -m elf_i386 -o task8 task8.o

(cmutale@cmutale)-[~/work/arch-pc/lab08]
$ ./task8 1 2 3
Функция:  $f(x)=5(2+x)$ 
Результат: 60
```

Рис 24

5 Выводы

При выполнении данной работы я освоила использование циклов и обработку аргументов командной строки.

6 Список литературы

Архитектура ЭВМ