Отчёта по лабораторной работе №8

**Команды безусловного и условного переходов в Nasm.
Программирование ветвлений.**

Газизянов Владислав Альбертович

Содержание

Список иллюстраций

1 Цель работы

Изучить работу циклов и обработкой аргументов командной строки.

2 Задание

Написать программы с использованием циклов и обработкой аргументов командной строки.

3 Выполнение лабораторной работы

Создаем каталог для программ ЛБ8, и в нем создаем файл

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

Рис. 3.1: Создание каталога

Заполняем его в соответствии с листингом 8.1

```
1  ;-------------------------------------------------------------------
2  ; Программа вывода значений регистра 'ecx'
3  ;-------------------------------------------------------------------
4  %include 'in_out.asm'
5  SECTION .data
6  msg1 db 'Введите N: ',0h
7  SECTION .bss
8  N: resb 10
9  SECTION .text
10 global _start
11 _start:
12 ; ----- Вывод сообщения 'Введите N: '
13 mov eax,msg1
14 call sprint
15 ; ----- Ввод 'N'
16 mov ecx, N
17 mov edx, 10
18 call sread
19 ; ----- Преобразование 'N' из символа в число
20 mov eax,N
21 call atoi
22 mov [N],eax
23 ; ------ Организация цикла
24 mov ecx,[N] ; Счетчик цикла, `ecx=N`
25 label:
26 mov [N],ecx
27 mov eax,[N]
28 call iprintLF ; Вывод значения `N`
29 loop label ; `ecx=ecx-1` и если `ecx` не '0'
30 ; переход на `label`
31 call quit
```

Рис. 3.2: Заплнение 8.1

Создаем исполняемый файл и запускаем его

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

Рис. 3.3: Проверка

Снова открываем файл для редактирования и изменяем его, добавив изменение значения регистра в цикле

```
label:
sub ecx,1 ; `ecx=ecx-1`
mov [N],ecx
mov eax,[N]
call iprintLF
loop label
```

Рис. 3.4: Вносим изменения

Создаем исполняемый файл и запускаем его

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

Рис. 3.5: Проверка

Регистр ecx принимает значения, в цикле label данный регистр уменьшается на 2 командой sub и loop). Число проходов цикла не соответсвует числу N, так

как уменьшается на 2. Снова открываем файл для редактирования и изменяем его, чтобы все корректно работало

```
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. 3.6: Редактирование

Создаем исполняемый файл и запускаем его

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

Рис. 3.7: Проверка работы

В данном случае число проходов цикла равна числу N.

Создаем новый файл

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

Рис. 3.8: Новый файл

Заполняем его в соответствии с листингом 8.2

```
%include 'in_out.asm'
SECTION .text
global _start
_start:
pop ecx ; Извлекаем из стека в `ecx` количеств
; аргументов (первое значение в стеке)
pop edx ; Извлекаем из стека в `edx` имя програ
; (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. 3.9: Заполняем 8.2

Создаем исполняемый файл и проверяем его работу, указав аргументы

```
vagazizyanov@vagazizyanov:~/work/arch-pc/lab08$ nasm -f elf lab8-2.asm
vagazizyanov@vagazizyanov:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-2 lab8-2.
o
vagazizyanov@vagazizyanov:~/work/arch-pc/lab08$ ./lab8-2
vagazizyanov@vagazizyanov:~/work/arch-pc/lab08$ ./lab8-2 2 3 '4'
2
3
4
```

Рис. 3.10: Проверка

Програмой было обработано 3 аргумента.

Создаем новый файл lab8-3.asm

```
vagazizyanov@vagazizyanov:~/work/arch-pc/lab08$ touch lab8-3.asm
```

Рис. 3.11: Новый файл

Открываем файл и заполняем его в соответствии с листингом 8.3

```
%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. 3.12: Заполнение 8.3

Создаём исполняемый файл и запускаем его, указав аргументы

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

Рис. 3.13: Проверка

Снова открываем файл для редактирования и изменяем его, чтобы вычислялось произведение вводимых значений

```
%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
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. 3.14: Редактирование

Создаём исполняемый файл и запускаем его, указав аргументы

```
vagazizyanov@vagazizyanov:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
vagazizyanov@vagazizyanov:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.
o
vagazizyanov@vagazizyanov:~/work/arch-pc/lab08$ ./lab8-3 12 13 7 10 5
Результат: 54600
```

Рис. 3.15: Проверка работы

3.1 Самостоятельная работа

Вариант 2 Напишите программу, которая находит сумму значений функции f(x) для x = x1, x2, …, xn, т.е. программа должна выводить значение f(x1) + f(x2) + … + f(xn). Значения xi передаются как аргументы. Вид функции f(x) выбрать из

таблицы 8.1 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу на нескольких наборах x = x1, x2, …, xn. Создаем новый файл

```
vagazizyanov@vagazizyanov:~/work/arch-pc/lab08$ touch lab8-4.asm
```

Рис. 3.16: Новый файл

Открываем его и пишем программу, которая выведет сумму значений, получившихся после решения выражения 3x-1

```
%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
mov ebx,3
mul ebx
sub eax,1
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. 3.17: Пишем программу

Транслируем файл и смотрим на работу программы

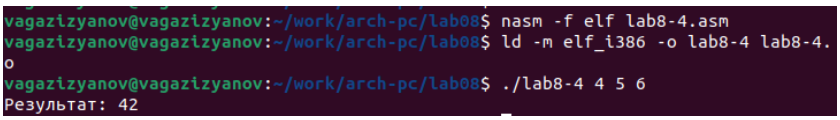

Рис. 3.18: Проверка

4 Выводы

Мы научились решать программы с использованием циклов и обработкой аргументов командной строки.