Лабораторная работа №8

Воинов Кирилл Викторович

Содержание

Список иллюстраций

1 Цель работы

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

2 Выполнение лабораторной работы

1. Создаю каталог для программ лабораторной работы No 8, перехожу в него и создаю файл lab8-1.asm.(рис. 2.1)

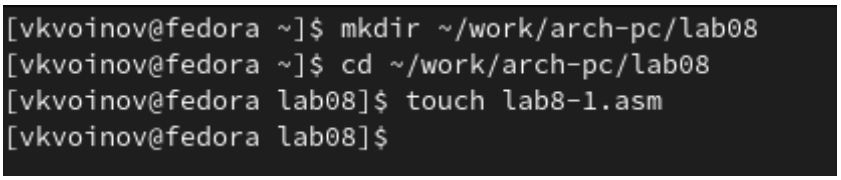

Рис. 2.1: Создание каталога, переход в него и создание файла

Ввожу в файл lab8-1.asm текст программы из листинга 8.1. (рис. 2.2)

```
;-------------------------------------------------------------------
; 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. 2.2: Текст программы

Создаю исполняемый файл и проверяю его работу.(рис. 2.3)

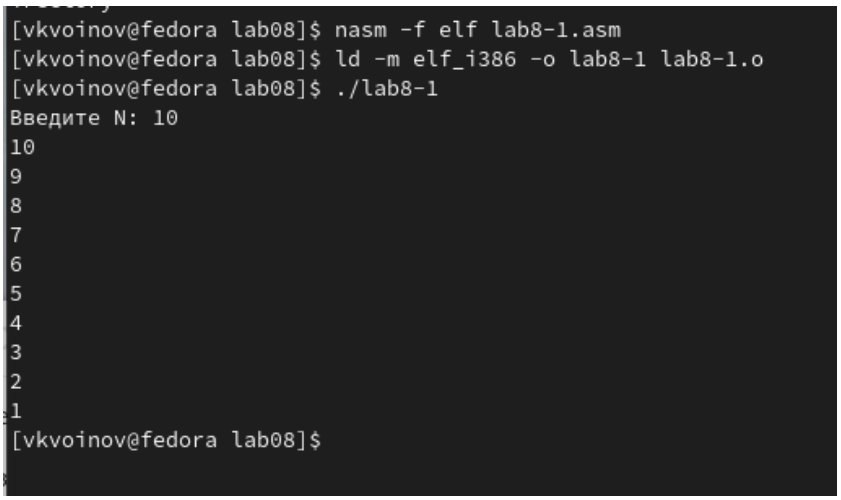

Рис. 2.3: Работа программы

Изменяю текст программы добавив изменение значения регистра ecx в цикле.(рис. 2.4)

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

Рис. 2.4: Измененный текст программы

Создаю исполняемый файл и проверяю его работу.(рис. 2.5)

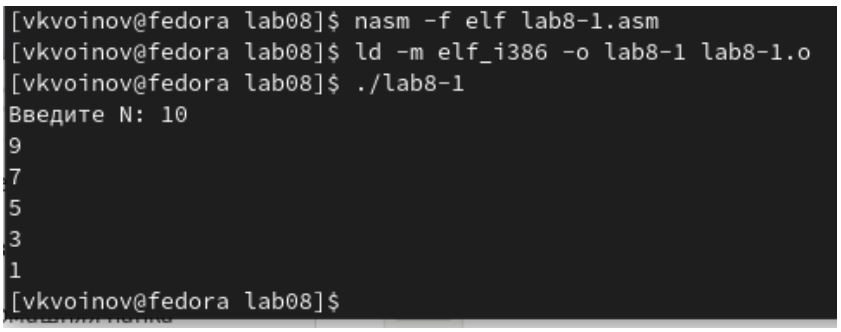

Рис. 2.5: Работа измененной программы

Регистр ecx принимает значения 10,9,7,5,3,1,0. Число проходов цикла не соответствует значению N введенному с клавиатуры.

Вношу изменения в текст программы добавив команды push и pop (добавления в стек и извлечения из стека) для сохранения значения счетчика цикла loop.(рис. 2.6)

```
;-------------------------------------------------------------------
; 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
; переход на `label`
call quit
```

Рис. 2.6: Измененный текст программы

Создаю исполняемый файл и проверяю его работу.(рис. 2.7)

```
[vkvoinov@fedora lab08]$ nasm -f elf lab8-1.asm
[vkvoinov@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[vkvoinov@fedora lab08]$ ./lab8-1
Введите N: 10
9
8
7
6
5
4
3
2
1
0
[vkvoinov@fedora lab08]$
```

Рис. 2.7: Работа измененной программы

Число проходов цикла соответствует значению N введенному с клавиатуры.

2. Создаю файл lab8-2.asm в каталоге ~/work/arch-pc/lab08.(рис. 2.8)

```
[vkvoinov@fedora lab08]$ touch lab8-2.asm
[vkvoinov@fedora lab08]$
```

Рис. 2.8: Создание файла

Ввожу в файл lab8-2.asm текст программы из листинга 8.2.(рис. 2.9)

```
;-------------------------------------------------------------------
; 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 sprintLF ; вызываем функцию печати
loop next ; переход к обработке следующего
; аргумента (переход на метку `next`)
_end:
call quit
```

Рис. 2.9: Текст программы

Создаю исполняемый файл и проверяю его работу, указав аргументы: аргумент1 аргумент 2 ‘аргумент 3’.(рис. 2.10)

```
[vkvoinov@fedora lab08]$ nasm -f elf lab8-2.asm
[vkvoinov@fedora lab08]$ ld -m elf_i386 -o lab8-2 lab8-2.o
[vkvoinov@fedora lab08]$ ./lab8-2 аргумент1 аргумент 2 'аргумент 3'
аргумент1
аргумент
2
аргумент 3
[vkvoinov@fedora lab08]$
```

Рис. 2.10: Работа программы

Было обработано программой 4 аргумента.

Создаю файл lab8-3.asm в каталоге ~/work/arch-pc/lab08.(рис. 2.11)

```
[vkvoinov@fedora lab08]$ touch lab8-3.asm
[vkvoinov@fedora lab08]$
```

Рис. 2.11: Создание файла

Ввожу в него текст программы из листинга 8.3.(рис. 2.12)

```
%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. 2.12: Текст программы

Создаю исполняемый файл и запускаю его, указав аргументы.(рис. 2.13)

```
[vkvoinov@fedora lab08]$ nasm -f elf lab8-3.asm
[vkvoinov@fedora lab08]$ ld -m elf_i386 -o lab8-3 lab8-3.o
[vkvoinov@fedora lab08]$ ./lab8-3 12 13 7 10 5
Результат: 47
[vkvoinov@fedora lab08]$
```

Рис. 2.13: Работа программы

Изменяю текст программы из листинга 8.3 для вычисления произведения аргументов командной строки.(рис. 2.14)

```
%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
; 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. 2.14: Текст измененной программы

Создаю исполняемый файл и запускаю его, указав аргументы.(рис. 2.15)

```
[vkvoinov@fedora lab08]$ nasm -f elf lab8-3.asm
[vkvoinov@fedora lab08]$ ld -m elf_i386 -o lab8-3 lab8-3.o
[vkvoinov@fedora lab08]$ ./lab8-3 10 5 4
Результат: 200
[vkvoinov@fedora lab08]$ ./lab8-3 2 2 2 2
Результат: 16
[vkvoinov@fedora lab08]$
```

Рис. 2.15: Работа измененной программы

3 Задание для самостоятельной работы

Пишу программу, которая находит сумму значений функции f(x) для x=x1, x2,…, xn т.е. программа выводит значение f(x1) + f(x2) +…+f(xn). Значения xi передаются как аргументы. Вариант 18: f(x)=17+5x.(рис. 3.1)

```
%include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
msv db "f(x)=17+5x",0
SECTION .text
global _start
mov eax, msv
call iprintLF
_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`)
mov ebx,5
pop eax ; иначе извлекаем следующий аргумент из стека
call atoi ; преобразуем символ в число
mul ebx
add eax, 17
add esi, eax
; след. аргумент `esi=esi+f`
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. 3.1: Текст программы

Создаю исполняемый файл и проверяю его работу на нескольких наборах x=x1, x2,..., xn.(рис. 3.2)

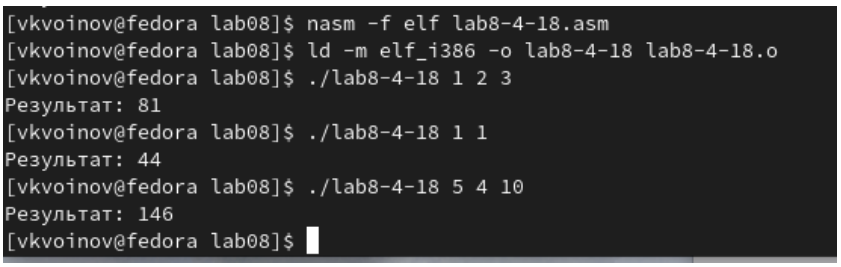

Рис. 3.2: Работа программы

4 Выводы

На этой лабораторной работе я приобрел навыки написания программ с использованием циклов и обработкой аргументов командной строки.