Отчёт по лабораторной работе №8

Дисциплина: Архитектура Компьютера

Иванов Сергей Владимирович

Содержание

Список иллюстраций

1 Цель работы

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

2 Выполнение лабораторной работы

Создадим каталог для программ лабораторной работы № 8, перейдем в него и создадим файл lab8-1.asm.(Рис. 2.1)

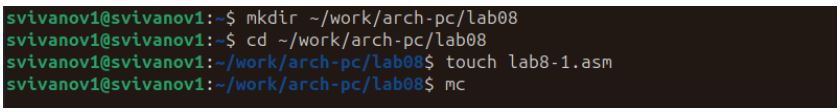

Рис. 2.1: Создание lab8-1.asm

Введем в файл lab8-1.asm текст программы из листинга 8.1, создадим исполняемый файл и запустим его. (Рис. 2.2)

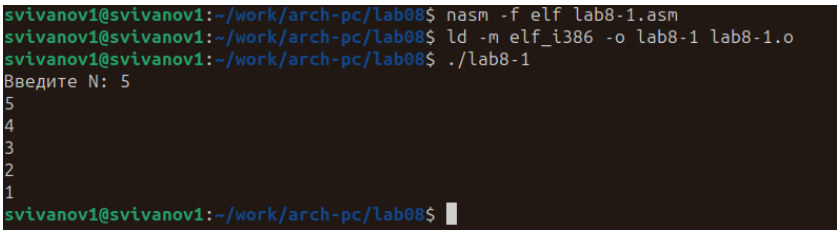

Рис. 2.2: Программа из листинга 8.1

Изменим текст программы добавив изменение значение регистра ecx в цикле, добавим строчку sub ecx,1 , создадим исполняемый файл и запустим его. (Рис. 2.3)

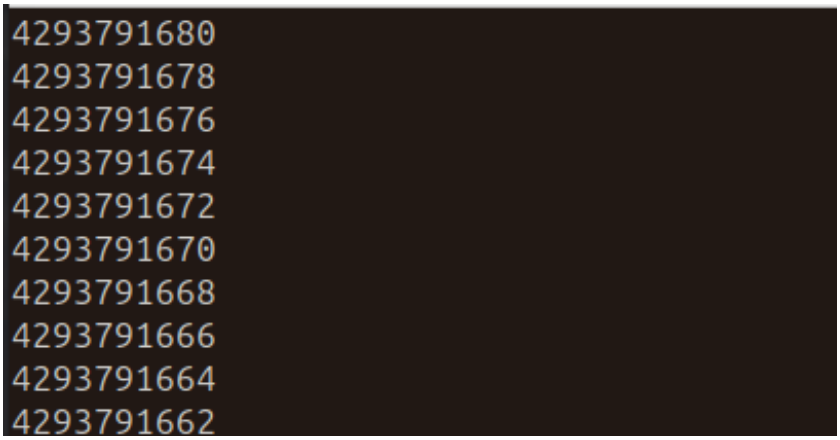

Рис. 2.3: Измененный код

Каждую итерацию ecx принимает значения, на 2 меньшее предыдущего. Из-за этого, количество проходов в 2 раза меньше, чем N. При этом, если N нечётное, ecx не сможет достичь нуля (т.к. мы через него перепрыгиваем, от 1 до -1), из-за чего получается бесконечный цикл

Внесём изменения в текст программы добавив команды push и pop (добавления в стек и извлечения из стека) для сохранения значения счетчика цикла loop, создадим исполняемый файл и запустим его. (Рис. 2.4)

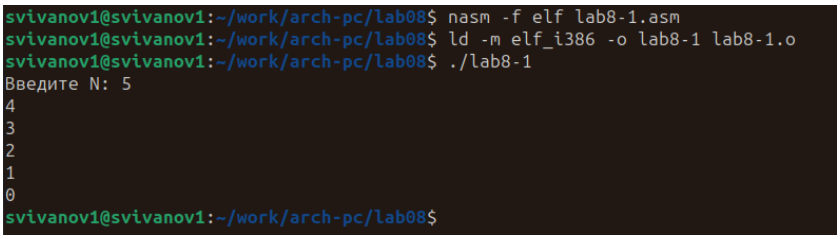

Рис. 2.4: Добавление push и pop

Теперь число проходов цикла соответствует значению N введенному с клавиатуры.

Создадим файл lab8-2.asm в каталоге ~/work/arch-pc/lab08 и введём в него текст программы из листинга 8.2. Создадим исполняемый файл и запустим его, указав аргументы. (Рис. 2.5)

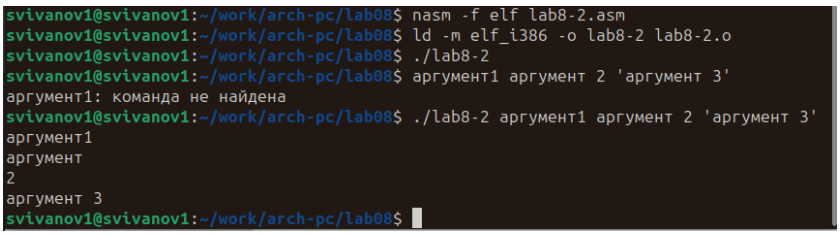

Рис. 2.5: Программа lab8-2.asm

Программа обработала 4 аргумента.

Создадим файл lab8-3.asm в каталоге ~/work/arch-pc/lab08 и введём в него текст программы из листинга 8.3. Создадим исполняемый файл и запустим его указав аргументы. (Рис. 2.6)

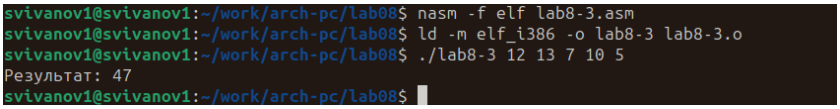

Рис. 2.6: Файл lab8-3.asm

Изменим текст программы из листинга 8.3 для вычисления произведения аргументов командной строки. (Рис. 2.7)

```
_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: "
```

Рис. 2.7: Изменяем программу

Создадим исполняемый файл и запустим его. (Рис. 2.8)

```
svivanov1@svivanov1:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
svivanov1@svivanov1:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
svivanov1@svivanov1:~/work/arch-pc/lab08$ ./lab8-3 12 13 7 10 5
Результат: 54600
svivanov1@svivanov1:~/work/arch-pc/lab08$
```

Рис. 2.8: Файл листинга

Напишем программу lab8-4.asm, которая находит сумму значений функции f(x) для варината 8 (f(x)=7+2x). (Рис. 2.9)

```
svivanov1@svivanov1:~/work/arch-pc/lab08$ ./lab8-4 10 11
Функция: f(x)=7+2x
Результат: 56
svivanov1@svivanov1:~/work/arch-pc/lab08$ ./lab8-4 1 2 3 4
Функция: f(x)=7+2x
Результат: 48
svivanov1@svivanov1:~/work/arch-pc/lab08$ ./lab8-4 10 11 12
Функция: f(x)=7+2x
Результат: 87
svivanov1@svivanov1:~/work/arch-pc/lab08$
```

Рис. 2.9: Создаем программу

Листинг программы для самостоятельной работы (lab8-4.asm):

```
%include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
msg1 db "Функция: f(x)=7+2x",0
SECTION .text
global _start
_start:
mov eax,msg1
call sprintLF
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 eax,eax
add eax,7
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
```

3 Выводы

В результате выполнения лабораторной работы мы приобрели навыки написания программ с использованием циклов и обработки аргументов командной строки.