Отчёт по лабораторной работе №8

Программирование цикла. Обработка аргументов командной строки.

Попутников Егор Сергеевич

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

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

2 Задание

1. 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.

3 Выполнение лабораторной работы

Создадим каталог для программ лабораторной работы № 8, перейдем в него и создадим файл lab8-1.asm:(3.1)

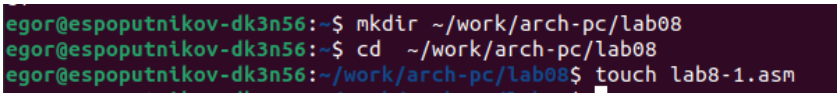

Рис. 3.1: Создание файла

Введем в файл lab8-1.asm текст программы из листинга 8.1. Создадим исполняемый файл и проверим его работу.(3.2)

Рис. 3.2: Создание исполняемого файла

Данный пример показывает, что использование регистра ecx в теле цилка loop может привести к некорректной работе программы. Изменим текст программы добавив изменение значение регистра ecx в цикле:(3.3)

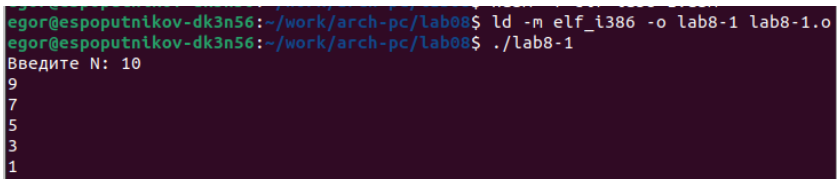

Рис. 3.3: Создание исполняемого файла

Регистр ecx принимает нечётные значения,число проходов цикла не соответствует значению N.

Для использования регистра ecx в цикле и сохранения корректности работы программы можно использовать стек. Внесем изменения в текст программы добавив команды push и pop (добавления в стек и извлечения из стека) для сохранения значения счетчика цикла loop:(3.4)

Рис. 3.4: Создание файла листинга

В данном случае число проходов цикла соответствует значению N.

При разработке программ иногда встает необходимость указывать аргументы, которые будут использоваться в программе, непосредственно из командной строки при запуске программы. При запуске программы в NASM аргументы командной строки загружаются в стек в обрат- ном порядке, кроме того в стек записывается имя программы и общее количество аргументов. Последние два элемента стека для программы, скомпилированной NASM, – это всегда имя программы и количество переданных аргументов. Таким образом, для того чтобы использовать аргументы в программе, их просто нужно извлечь из стека. Обработку аргументов нужно проводить в цикле. Т.е. сначала нужно из- влечь из стека

количество аргументов, а затем циклично для каждого аргумента выполнить логику программы.Создадим файл lab8-2.asm в каталоге ~/work/arch-pc/lab08 и введем в него текст программы из листинга 8.2. Создадим исполняемый файл и запустим его, указав аргументы:(3.5)

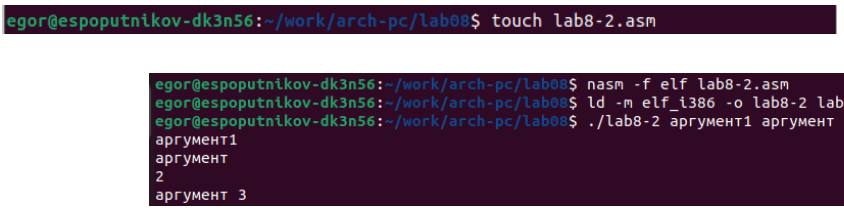

Рис. 3.5: Создание исполняемого файла

Программой было обработано 4 аргумента.

Создадим файл lab8-3.asm в каталоге ~/work/arch- pc/lab08 и введем в него текст программы, которая выводит сумму чисел, которые пере- даются в программу как аргументы.(3.6)

Рис. 3.6: Программа вывода суммы

Изменим текст программы из листинга 8.3 для вычисления произведения аргументов командной строки.(3.7)

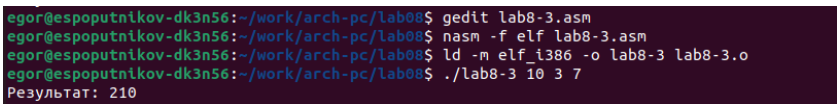

Рис. 3.7: Программа вывода произведения

4 Выполнение самостоятельной работы

1. 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.Вариант №4.(4.1)

```
%include 'in_out.asm'
SECTION .data
msg db "Результат: ", 0
SECTION .bss
prm: RESB 80
SECTION .text
global _start
_start:
pop ecx
pop edx
sub ecx, 1
mov esi, 2

next:
cmp ecx, 0
jz _end
pop eax
call atoi
mul esi
sub eax,2
add [prm],eax
loop next
_end:
mov eax,msg
call sprint
mov eax,[prm]
call iprintLF
call quit

```

```
egor@espoputnikov-dk3n56:~/work/arch-pc/lab08$ nasm -f elf variant4.asm
egor@espoputnikov-dk3n56:~/work/arch-pc/lab08$ ld -m elf_i386 -o variant4 varian
t4.o
egor@espoputnikov-dk3n56:~/work/arch-pc/lab08$ ./variant4 1 2 3 4
Результат: 12
egor@espoputnikov-dk3n56:~/work/arch-pc/lab08$
```

Рис. 4.1: Самостоятельная работа

5 Выводы

Я приобрел навыки написания программ с использованием циклов и обработкой аргументов командной строки.