Лабораторная работа №8

Программирование цикла. Обработка аргументов командной строки

Яковлева Дарья Сергеевна

Содержание

Список иллюстраций

Список таблиц

# 1 Цель работы

Научиться работать с циклами на языке Ассемблера, а также научиться обрабатывать аргументы командной строки

# 2 Выполнение лабораторной работы

Для начала выполнения лабораторной работы создадим рабочую директорию и файл lab8-1.asm (рис. 2.1):

|  |
| --- |
| Создание рабочей директории и файла lab8-1.asm |

Создание рабочей директории и файла lab8-1.asm

Далее, запустим Midnight commander (рис. 2.2):

|  |
| --- |
| Запуск Midnight commander |

Запуск Midnight commander

Теперь, вставим в ранее созданный файл из листинга 8.1. Он должен запускать цикл и выводить каждую итерацию число, на единицу меньше предыдущего (начинается выводить с числа N) (рис. 2.3):

|  |
| --- |
| Вставка кода из файла листинга 8.1 |

Вставка кода из файла листинга 8.1

Чтобы собрать код, нужен файл in\_out.asm. скопируем его из директории прошлой лабораторной работы (рис. 2.4):

|  |
| --- |
| Копирование файла in\_out.asm в рабочую директорию |

Копирование файла in\_out.asm в рабочую директорию

Теперь соберём программу и посмотрим на результат выполнения (рис. 2.5):

|  |
| --- |
| Сборка программы из файла lab8-1.asm и её запуск |

Сборка программы из файла lab8-1.asm и её запуск

Как видим, она выводит числа он N до единицы включительно. Теперь попробуем изменить код, чтобы в цикле также отнималась единица у регистра ecx (рис. 2.6):

|  |
| --- |
| Изменение файла lab8-1.asm |

Изменение файла lab8-1.asm

Попробуем собрать программу и запустить её (рис. 2.7):

|  |
| --- |
| Повторная сборка программы из файла lab8-1.asm и её запуск |

Повторная сборка программы из файла lab8-1.asm и её запуск

Введём в качестве N число 5 и посмотрим на результат выполнения (рис. 2.8):

|  |
| --- |
| Результат вывода |

Результат вывода

Как видим, цикл выполняется бесконечное количество раз. Это связано с тем, что цикл останавливается в тот момент, когда при проверке ecx равен 0, но он каждое выполнение цикла уменьшается на 2, из-за чего, в случае нечётного числа, никогда не достигнет нуля. Регистр ecx меняет своё значение дважды: стандартно -1 после каждой итерации и -1 в теле цикла из-за команды sub. Если на вход подать чётное число, цикл прогонится N/2 раз, выводя числа от N-1 до 1 (выводит через одно число) (рис. 2.9):

|  |
| --- |
| Результат вывода для чётного N |

Результат вывода для чётного N

Таким образом, количество итераций цикла не равно N ни при подаче на вход чётного числа, ни при подаче нечётного.

Теперь попробуем изменить программу так, чтобы она сохраняла значение регистра ecx в стек (рис. 2.10):

|  |
| --- |
| Редактирование файла lab8-1.asm |

Редактирование файла lab8-1.asm

Попробуем собрать и запустить программу (рис. 2.11):

|  |
| --- |
| Повторная сборка программы из файла lab8-1.asm и её запуск |

Повторная сборка программы из файла lab8-1.asm и её запуск

Теперь, программа выводит все числа от N-1 до нуля. Таким образом, число прогонов цикла равно числу N. Создадим второй файл (рис. 2.12):

|  |
| --- |
| Создание второго файла: lab8-2.asm |

Создание второго файла: lab8-2.asm

И вставим в него код из файла листинга 8.2 (рис. 2.13):

|  |
| --- |
| Запись кода из листинга 8.2 в файл lab8-2.asm |

Запись кода из листинга 8.2 в файл lab8-2.asm

Соберём и запустим его, указав некоторые аргументы. Посмотрим на результат (рис. 2.14):

|  |
| --- |
| Сборка программы из файла lab8-2.asm и её запуск |

Сборка программы из файла lab8-2.asm и её запуск

Как видим, он обработал 4 аргумента. Аргументы разделяются пробелом, либо, когда аргумент содержит в себе пробел, обрамляется в кавычки. Создадим третий файл (рис. 2.15):

|  |
| --- |
| Создание третьего файла: lab8-3.asm |

Создание третьего файла: lab8-3.asm

И вставим в него код из листинга 8.3. Он будет находить сумму всех аргументов (рис. 2.16):

|  |
| --- |
| Запись кода из листинга 8.3 в файл lab8-3.asm |

Запись кода из листинга 8.3 в файл lab8-3.asm

Теперь соберём программу и запустим её (рис. 2.17):

|  |
| --- |
| Сборка программы из файла lab8-2.asm и её запуск |

Сборка программы из файла lab8-2.asm и её запуск

Как видим, программа действительно выводит сумму всех аргументов. Изменим её так, чтобы она находила не сумму, а произведение всех аргументов (рис. 2.18):

|  |
| --- |
| Изменение файла lab8-3.asm |

Изменение файла lab8-3.asm

Соберём программу и запустим её (рис. 2.19):

|  |
| --- |
| Повторная сборка программы из файла lab8-3.asm и её запуск |

Повторная сборка программы из файла lab8-3.asm и её запуск

Как видим, программа выводит правильный ответ

# 3 Выполнение задания для самостоятельной работы

Для выполнения самостоятельной работы создадим файл в формате .asm (рис. 3.1):

|  |
| --- |
| Создание файла самостоятельной работы |

Создание файла самостоятельной работы

В рамках самостоятельной работы необходимо сделать задание под вариантом 17. Так, необходимо сложить результаты выполнения функции f(x)=10(𝑥 − 1) для всех введённых аргументов (рис. 3.2):

|  |
| --- |
| Код файла самостоятельной работы |

Код файла самостоятельной работы

Соберём и запустим программу, вводя различные аргументы (рис. 3.3):

|  |
| --- |
| Сборка и запуск программы первого задания самостоятельной работы, а также результат выполнения |

Сборка и запуск программы первого задания самостоятельной работы, а также результат выполнения

Пересчитав результат вручную, убеждаемся, что программа работает верно

# 4 Выводы

В результате выполнения лабораторной работы были получены навыки работы с циклами и обработкой аргументов из командной строки. Были написаны программы, использующие все вышеописанные аспекты.