Отчёт по ИДЗ №2

ФИО: Сайфутдинов Рафаэль Рустамович

Группа: БПИ216

Вариант задания: 10

Условие задания:

10. Разработать программу, которая меняет на обратный порядок следования символов **каждого слова** в ASCII-строке символов. Порядок слов остается неизменным. Слова состоят только из букв. Разделителями слов являются все прочие символы.

Для удобства буквами будем считать символы латинского алфавита в нижнем и верхнем регистрах. Все прочие символы (в том числе символы русского алфавита) будут считаться разделителями слов.

Тесты, демонстрирующие проверку программ:
Подготовлено 5 тестов, покрывающих достаточное количество возможных результатов работы программы.
На вход подаётся строка длиной не более 100 символов. Каждое слово может быть не длиннее 50 символов.

Тест №1:
Команда: ./main.exe < Tests/test1.in
Входные данные: Hello, world!
Ожидаемые выходные данные: olleH, dlrow!
Фактические выходные данные: olleH, dlrow!

Тест №2:
Команда: ./main.exe < Tests/test2.in
Входные данные: 12345
Ожидаемые выходные данные: 12345
Фактические выходные данные: 12345
Тест №3:
Команда: ./main.exe < Tests/test3.in
Входные данные: rEvErSeD
Ожидаемые выходные данные: DeSrEvEr
Фактические выходные данные: DeSrEvEr

Тест №4:
Команда: ./main.exe < Tests/test4.in
Входные данные: words1divided2by3numbers
Ожидаемые выходные данные: sdrow1dedivid2yb3srebmun
Фактические выходные данные: sdrow1dedivid2yb3srebmun

Тест №5:
Команда: ./main.exe < Tests/test5.in
Входные данные: ABC_def_G#HiJk
Ожидаемые выходные данные: CBA_fed_G#kJiH
Фактические выходные данные: CBA_fed_G#kJiH

Исходный текст программы на языке С можно посмотреть, открыв файл main.c
Текст финальной программы на языке ассемблера, полученной после компиляции и с расширенными комментариями и оптимизацией можно посмотреть, открыв файл main.S
Текст программы на языке ассемблера, полученной после компиляции и с расширенными комментариями без оптимизации можно посмотреть, открыв файл main_without_optimization.S
Текст программы на ассемблере без комментариев, полученный сразу после компиляции программы на языке С, можно посмотреть, открыв файл main_before.S
(Я не стал вставлять сюда тексты, чтобы не «захламлять» PDF файл)

Информация, подтверждающая выполнение задания в соответствии требованиями на **оценку 6:**

- Программа на ассемблере была скомпилирована при помощи соответствующих **аргументов командной строки** (gcc -S -masm=intel -fno-asynchronous-unwind-tables -fno-jump-tables -fno-stack-protector -fno-exceptions main.c -o main.s), а также за счёт **ручного** редактирования исходного текста ассемблерного кода программы;
- В реализованной программе использованы функции revstr и printChangedString с передачей данных через параметры;
- Добавлены исчерпывающие **комментарии**, поясняющие эквивалентное представление переменных в программе на С, а также описывающие передачу фактических параметров и перенос возвращаемого результата в функциях;
- Представлен **набор тестов**, на которых **корректно** отработала программа как на языке С, так и на ассемблере;
- Размер объектного файла на ассемблере, полученного после компиляции на С (16248 бит), **больше** объектного файла модифицированной программы, использующей регистры (16096 бит), на **152 бита**
- Максимально использованы регистры процессора вместо локальных переменных (и не только), чтобы увеличить производительность. Подробнее:
    - **Убраны** команды nop, которые буквально «ничего не делают»;
    - Операция «зануления» регистра **оптимизирована** при помощи xor;
    - **Использование** «нестираемого» 4-х байтового **регистра** r12d вместо локальной переменной -4[rbp] (для переменной i в функции revstr в коде на С)
    - **Использование** «нестираемого» 4-х байтового **регистра** r13d вместо локальной переменной -20[rbp] (для переменной i в функции printChangedString коде на С)

- **Использование** «нестираемого» 4-х байтового **регистра** r14d вместо локальной переменной -8[rbp] (для переменной len в функции revstr в коде на C), однако для перемещения все-таки один раз -8[rbp] должен использоваться с командой mov;
- **Удаление** ненужных операций mov для перемещения значения из одного регистра на другой, которые используют регистр eax в качестве «посредника» (помечено комментарием «Оптимизация»)