Министерство образования Республики Беларусь
Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей
Кафедра Информатики
Дисциплина «Конструирование программ»

ОТЧЕТ
к лабораторной работе №3
на тему:
«Создание простой программы на языке ассемблера. Обработка символьных данных»
БГУИР 6-05-0612-02 49

Выполнил студент группы 353502
ЗГИРСКАЯ Дарья Денисовна

(дата, подпись студента)

Проверил ассистент каф. Проектирования информационно-компьютерных систем
СМОРГУН Евгений Святославович

(дата, подпись преподавателя)

Минск 2024

1 ЗАДАНИЕ

Задание 1. Вариант 1.

Написать программу «Hello, world!».

Задание 2. Вариант 15.

Вставить в строке слово «number» перед словами, состоящими только из цифр.

2 ВЫПОЛНЕНИЕ РАБОТЫ

Перед началом выполнения работы был изучен теоретический материал лабораторного практикума из разделов «Создание простой программы на языке ассемблера» и «Обработка симовльных данных».

Целью первого создания было создание программы, выводящей в консоль строку «Hello, world!». Сделать это надо было в двух форматах исходного файла: .com и .exe.

В перовм случае код не содержит дополнительную информацию о программе, а три сегмента: сегмент кода, данных и стека – объединены в один сегмент, который занимает не больше 64 Кбайт. Чтобы вывести строку в консоль, необходимо для начала задать саму строку. Это было выполнено определение последовательности байт с помощью ключевого слова db (define byte). В конце строки был добавлен символ ‘$’, который обозначает ее конец (рис. 1).

```
message db "Hello, World!$"
```

Рисунок 1 – Задание строки

Для того, чтобы вывести строку в консоль, необходимо воспользоваться прерыванием, которое действует слкдующим образом: необходимо занести его номер в DOS в регистр ah, занести адрес строки (относительно сегмента данных), которую нужно вывести, в регистр dx, после чего вызвать прерывание int 21h. Одним из вариантов вывода строки является прерывание 09h, что и было использовано (рис. 2).

```
mov ah, 9
mov dx, offset message
int 21h
```

Рисунок 2 – Вывод строки в консоль

Во втором случае сегменты данных, стека и программы отделяются друг от друга. Поэтому необходимо присвоить адрес сегмента .data, в котором хранится заданная строка, регистру ds (data segment). А в регистр dx (data) – адрес заданной строки относительно этого сегмента (рис. 3). После этого необходимо вызвать прерываение, что и в предыдущей программе, для вывода строки на в консоль.

```
mov ax, [data]
mov ds, ax

mov dx, offset message
```

Рисунок 3 – Присваивание адресов регистрам ds и dx

Для решения второго задания сначала необходимо создать буфер длиной в 201 байт (1 байт уйдет на то, чтобы записать, сколько байт из этого буфера было введено в консоли). Также необходимо создать строки, объявляющие пользователю о вводе строки и выводе обработанной строки, и строку, обозначающую переход на новую строку. Все это происходит в сегменте данных (рис. 4).

```
data segment
    write_string db "Write string (max character's number = 200): $"
    write_end_string db "New string: "
    new_line db 0x0A, 0x0D, '$'
    input_string db 201
                 db ?
                 db 201 dup(?)
ends
```

Рисунок 4 – Сегмент данных

Также необходимо задать сегмент стека (рис. 5).

```
stack segment
    db 200 dup(0)
ends
```

Рисунок 5 – Задание сегмента стека

Для решения задания можно использовать такую логику: ввести строку, записать ее в стек с конца, после чего доставать из стека по одному

байту и записывать в строку. После записи слова в строку, проверить, является ли записанное слово числом. Если не является, то просто продолжить записивать стек в строку. Если является, то необходиом записать это слово обратно в стек, записать в строку слово «number», после чего продолжить запись в строку из стека.

Для оптимизации написания кода можно создать три макроса: для вывода строки в консоль, для записи слова «number» в строку (рис. 6), а также для запсии строки в стек с добавлением символа «*» после занесения символа « » в стек. Последнее необходимо для того, чтобы при перенесениии символа «*» в строку записанное слово проверялось, является ли оно числом. На символе пробела регистру bx будет присваиваться 1 – это значит, что следующее за пробелом слово – число. Если же при переносе символов из стека в строку попадется символ, не являющийся цифрой, то регистру bx будет присвоен 0.

```
write_word macro
    mov [si], 'n'
    inc si
    mov [si], 'u'
    inc si
    mov [si], 'm'
    inc si
    mov [si], 'b'
    inc si
    mov [si], 'e'
    inc si
    mov [si], 'r'
    inc si
    mov [si], ' '
    add bp, 7
endm
```

Рисунок 6 – Макрос, записывающий в строку слово «number»

После завершения перезаписи строк остается только вывести в консоль символы новой строки и возврата каретки, подпись для пользователя «New string» и полученную в результате выполения программы строку.

ВЫВОД

В ходе лабораторной работы был изучен теоретический материал лабораторного практикума из разделов «Создание простой программы на языке ассемблера» и «Обработка симовльных данных». Было изучено разделение программы на сегменты данных, стека и кода. Были использованы

прерывания для ввода и вывода строки в консоль. Была написана программа, выводящая строку «Hello, world!» в консоль, причем в двух видах исходных файлов. Также была написана программа, использующая макросы (вывода строки, записи строки в стек, записи в строку слова «number»), имеющая в себе цикл перенесения данных из стека в строку с последующей проверкой записанного слова на число.