Оглавление

Описание задачи:

Задача: Разработать решение, на языке Ассемблер, для перевода входной строки формата ASCII в число формата short integer.

Техническое задание:

1. Входная строка задаётся динамически через консоль пользователем.
2. Отформатированные данные (т.е. число со знаком) записывается в особое место в памяти.
3. Строка может содержать как положительное, так и отрицательное число.
4. Дробные числа не встречаются.

Пример входных данных:	Число на выходе должно иметь вид:
-21	FFFFFFEB
-337	FFFFFEAF
334	014E

Описание решения:

Язык программирования: Ассемблер

Компилятор языка: Nasm x86[1]

Линковщик: GoLink[2]

Система: Windows 32 bit

Необходимые библиотеке на устройстве пользователя: user32.dll, kernel32.dll

Компилятор Nasm x86 выбран по следующим причинам:

1. Совместимость с целевой системой
2. Открытый исходный код
3. Простой синтаксис

Линковщик GoLink выбран по следующим причинам:

1. Открытый исходный код
2. Наличие документации

[1] Index of /pub/nasm/releasebuilds/3.01rc9/win32

[2] Go Tools for Windows (assembler, resource compiler, linker, debugger and information)

Система Windows 32 bit была выбрана по следующим причинам:

1. Доля рынка ноутбуков и персональных компьютеров у систем Windows составляет 71% (по данным википедии)[3]
2. Программы разработанные под 32 битную архитектуру корректно запускаются на системах 64 бита

Документация:

Полный код программы с метками (желтым цветом) приведен на странице .

Описание элементов кода по номерам меток (в коде желтым цветом):

1. Объявление дескриптора консоли.
2. Начало макроса input с нулем входных данных.
3. Enter 0,0 позволяет выполнить участок кода без изменения значения регистров и флагов.
4. Вызов функции чтения консоли.
5. Закрытие конструкции enter 0,0
6. Завершение макроса
7. Переменная, используемая для определения есть ли знак у числа.
8. Секция резервирования данных, без их объявления.
9. Резервирование пространства для отформатированного числа.
10. Метка старта.
11. extern подключает внешние функции. В этом случае: функция получения дескриптора, функция чтения консоли, функция завершения программы.
12. Само получения дескриптора консоли в начале программы.
13. Вызов макроса на чтение данных из консоли.
14. Проверка первого символа строки.
15. Метка показывающая, что первый символ строки (знак) пропущен.
16. Завершение программы.
17. Пропуск первого символа строки, если это минус.
18. Метка проверки, нужно ли добавлять к числу минус.
19. Метка, что со знаком все в порядке
20. Процедура перевода строки в число (работает только со строками, содержащими числа).

Такое большое количество меток обусловлено возможностью изменения решения под нужды конкретно проекта. Создав такой код, была оставлена возможность редактирования участков кода и добавления новых возможностей на разных этапах программы.

[3] Usage share of operating systems - Wikipedia

Код программы:

```
stdin equ -10 | 1
%macro input 0 | 2
    enter 0,0 |3
    push 0
    push InputLength
    push 256
    lea eax, [InputBuffer]
    push eax
    push dword [InputHandle]
    call _ReadFile@20 | 4
    leave | 5
%endmacro | 6
section .data
    sig db 0 | 7
section .bss | 8
    InputHandle resd 1
    InputBuffer resd 256
    InputLength resd 1
    Number resd 1 | 9
section .text
    global Start | 10
    extern _GetStdHandle@4, _ReadFile@20, _ExitProcess@4 | 11
Start:
    push stdin
    call _GetStdHandle@4
    mov dword [InputHandle], eax | 12
    input | 13
    mov edx, InputBuffer
    mov cl, [edx]
    cmp cl, '-' | 14
    je signed
    sign_deleted: | 15
    call string_to_number
    call num
    call _ExitProcess@4 | 16
signed: | 17
    inc eax
    inc edx
    inc [sig]
    jmp sign_deleted
num: | 18
    cmp [sig], 0
    je sign_ok
    neg eax
    sign_ok: | 19
    mov [Number], eax
    call _ExitProcess@4
string_to_number: | 20
    xor eax, eax
.top:
    movzx ecx, byte [edx]
    inc edx
    cmp ecx, '0'
    jb .done
    cmp ecx, '9'
    ja .done
    sub ecx, '0'
    mul eax, 10
    add eax, ecx
    jmp .top
.done:
    ret
```

Описание работы программы:

Программа получает дескриптор консоли и начинает считывать с неё данные. Пользователь вводит число (отрицательное или положительное). Код пропускает минус у строки, если он есть, и переводит строку в число. После этого число записывается в зарезервированное пространство в памяти (если у строки был минус, то число инвертируется и становится отрицательным).

Полный код программы занимает 1119 байт дискового пространства. Собранная программа для тестирования и отладки решения занимает 2560 байт дискового пространства.

Репозиторий: https://github.com/sowlKALI/lab2.git

Рисунок 1. QR-код ссылка на репозиторий с файлами.

Блок схема программы:

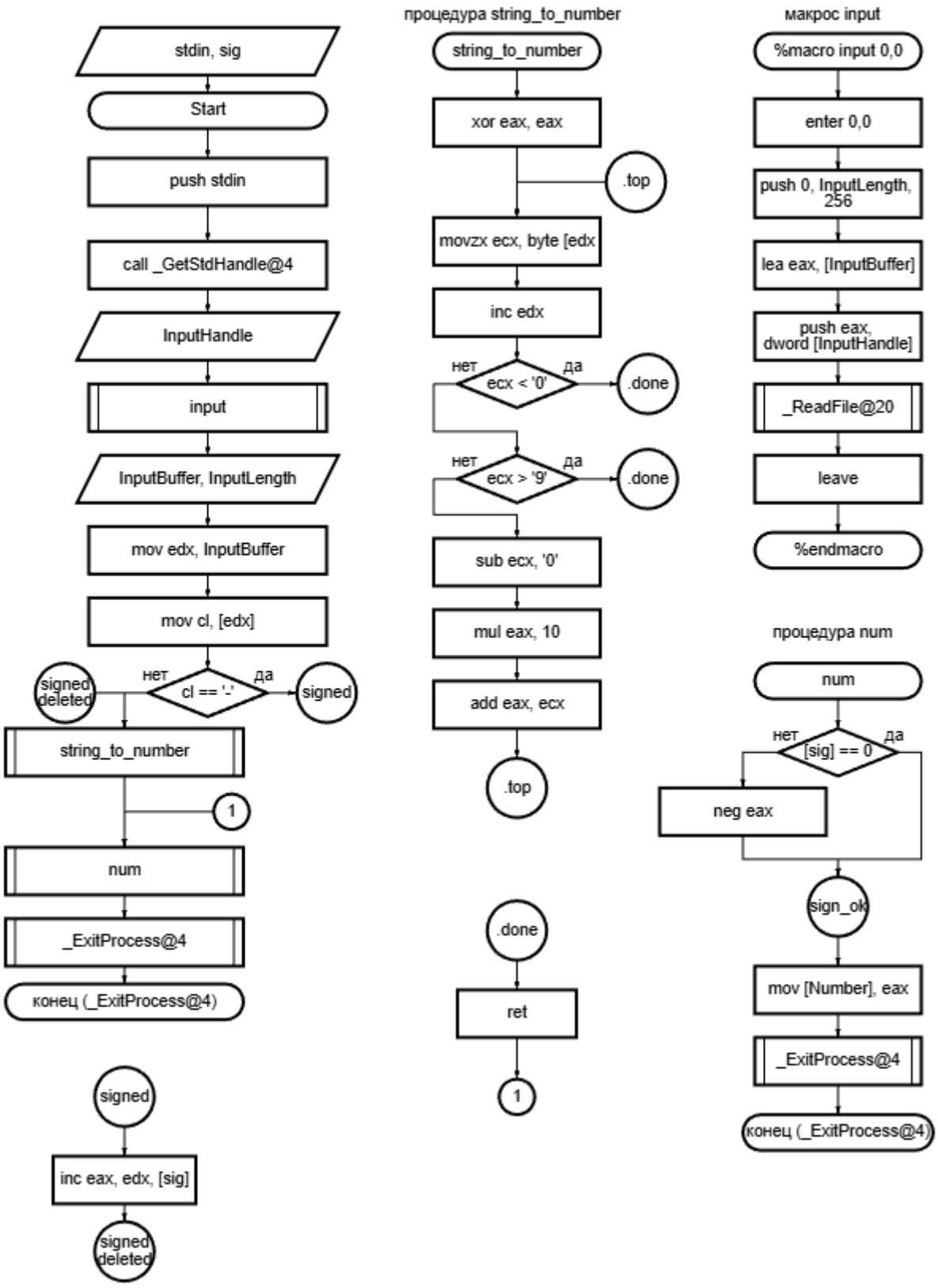

Рисунок 2. Блок схема программы.

Проверка работоспособности решения:

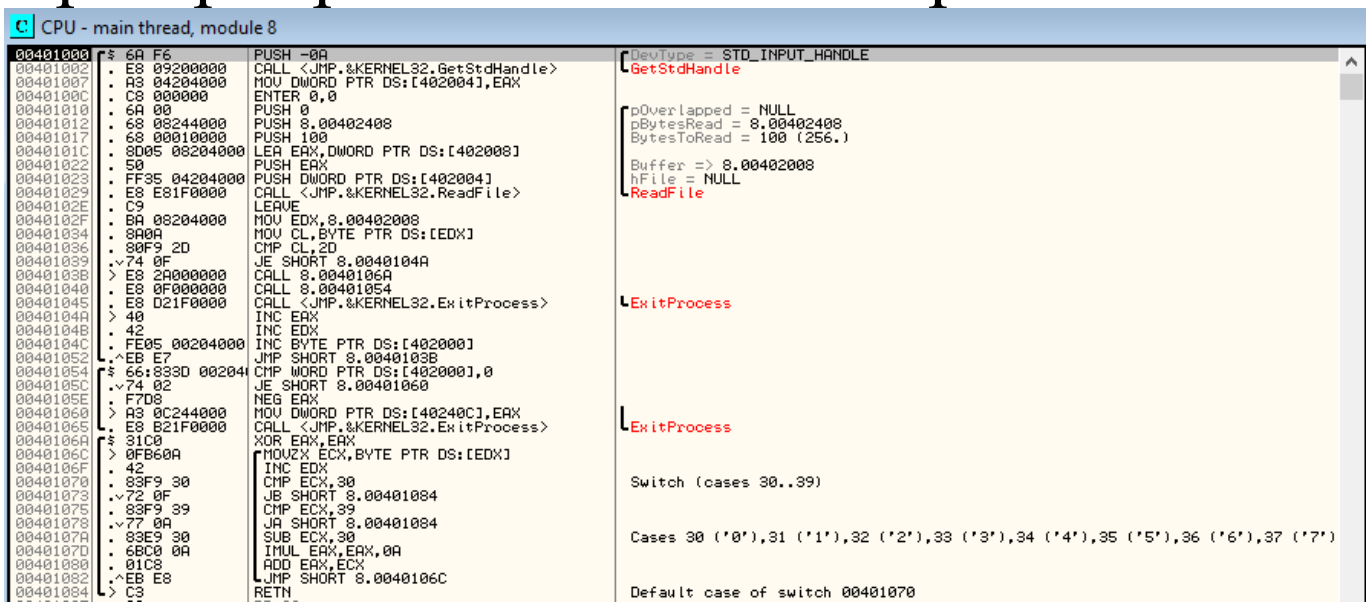

Рисунок 3. Снимок решения в отладчике.

Рисунок 4. Входная строка (вводится пользователем с клавиатуры).

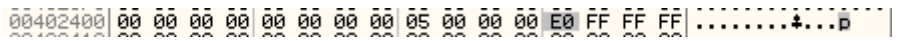

Рисунок 5. Дамп памяти, выделенной под хранение числа. (FFFFFFE0 = -32).

Таблица с тестовым набором:

Пример входных данных:	Число на выходе должно иметь вид:
-21	FFFFFFEB
-32	FFFFFFE0
-337	FFFFFEAF
334	014E

Требования к сборке решения:

1. Nasm x86 (рекомендованная версия: 3.01rc09)
2. GoLink (рекомендованная версия: 1.0.4.6)
3. Наличие стандартных библиотек Windows: user32.dll и kernel32.dll

Команды сборки:

1. *nasm -f win32 lab1.asm -o lab1.obj*
2. *golink /entry:Start /console kernel32.dll user32.dll lab1.obj*