|  |  |  |
| --- | --- | --- |
|  | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования «Новосибирский государственный технический университет» | | |
|  | | |
| Кафедра прикладной математики | | |
| Лабораторная работа № 1 | | |
| по дисциплине «Языки программирования и методы трансляции» | | |
| **Основные элементы языка ассемблера** | | |
|  | | |
|  | Группа ПМ-13 |  |
| Бригада 9 | Исакин даниил |
| Вариант 9 | вострецова екатерина |
|  |  |
|  |  |
| Преподаватель | сивак мария алексеевна |
|  | дворецкая виктория константиновна |
|  |  |
| Новосибирск, 2024 | | |

# Задание

1) ввод с клавиатуры 2-х чисел в заданной системе счисления;

2) выполнение арифметической операции над этими числами (в предположении, что размер чисел не вызывает переполнения регистров);

**Вариант 9:**

Умножение чисел, ввод в десятичной системе счисления (не менее 4-х знаков каждое число), вывод в шестнадцатеричной системе счисления

3) вывод результата в заданной системе счисления.

Все промежуточные данные должны сохраняться в памяти. При выводе результата не использовать функцию wsprintfA

# Алгоритм

• Считать первое число в виде строки

• Обработать первое число (преобразовать в число десятичной системы счисления):

Берем символы с конца строки

Последовательно обработать каждый символ:

Если символ от ‘0’ <= ch <= ‘9’  
 Вычесть из числа код ‘0’

Умножить число на коэффициент разряда. 10n

Добавить полученное число в результирующую переменную   
 Если символ ch = ‘-‘

Выставить бит знака

Иначе выдать ошибку и завершить программу;

• Считать второе число в виде строки

• Обработать первое число (преобразовать в число десятичной системы счисления):

Берем символы с конца строки

Последовательно обработать каждый символ:

Если символ от ‘0’ <= ch <= ‘9’  
 Вычесть из числа код ‘0’

Умножить число на коэффициент разряда. 10n

Добавить полученное число в результирующую переменную   
 Если символ ch = ‘-‘

Выставить бит знака

Иначе выдать ошибку и завершить программу;

• Вычислить бит знака результата

• Умножить числа

* Перевести результат в строку:

Загрузить бит знака.

Выставить бит знака в 0

В цикле по числу:

Берем младшие 8 бит

Применяем маску 0x0000000F – оставить 4 бита

Загрузить значение из таблицы по полученному адресу

Конец цикла

• Вывести результат

# Текст программы

.386

.MODEL FLAT, STDCALL

; прототипы внешних функций (процедур) описываются директивой EXTERN,

; после знака @ указывается общая длина передаваемых параметров,

; после двоеточия указывается тип внешнего объекта – процедура

EXTERN GetStdHandle@4: PROC

EXTERN WriteConsoleA@20: PROC

EXTERN CharToOemA@8: PROC

EXTERN ReadConsoleA@20: PROC

EXTERN ExitProcess@4: PROC

EXTERN lstrlenA@4: PROC

.CONST

mul\_coef dd 10 ; Коэффициент шага при переводе

.DATA ; data segment

din  dd ? ; Дескриптор ввода

dout dd ? ; Дескриптор вывода dd - резервирет память объемом 4 байта ? - используется для не

tmp\_local\_var\_byte db ? ; Локальная переменная для работы с символами

tmp\_local\_var dd ?      ; Локальная переменная, которая для чего нибудь может понадобиться

table\_char byte 30h, 31h, 32h, 33h, 34h, 35h, 36h, 37h, 38h, 39h, 41h, 42h, 43h, 44h, 45h, 46h

strn db "Введите строку число [минимум 4 знака числа]: ", 13, 10, 0; Строка для вывода

str\_overflow\_error\_msg db "Переполнение регистра при конвертации строки в число", 13, 10, 0

str\_overflow\_error\_msg\_mul db "Переполнение регистра при операции умножения", 13, 10, 0

str\_not\_correct\_num\_char db "Некоректный символ и/или запись числа. Допустимые символы: [0-9] и знак минуса", 13, 10, 0

str\_input\_str\_short db "Введенное число слишком короткое. Длина должна быть 4 символа и более", 13, 10, 0

outbuf db 12 dup(0); Размер буфера на выход

inbuf      db 15 dup(?) ; Размер буфера на вход программы

num1 dd 0 ; Первое число - знаковое

num2 dd 0 ; Второе число - знаковое

result dd 0 ; Результат умножения

.CODE

; Инициализация потока ввода данных

; Дескриптор будет сохранен в переменную din

OpenRead proc

    ; получить десериптор ввода

    push -10

    call GetStdHandle@4

    mov din, eax

    ret

OpenRead endp

; Получить дескриптор вывода данных

; Дескриптор будет сохранен в переменную dout

OpenWrite proc

    ; получить дескриптор выводы

    push -11

    call GetStdHandle@4

    mov dout, eax

    ret

OpenWrite endp

; Функция перекодировки строки для вывода информации

; eax - Адрес строки для перекодировки.

; Результат сохранен в переменную буфер, которая была передана в регистре EAX

ConvertToDOSStr proc

    push eax

    push eax

    call CharToOemA@8 ; Перекодировки

    ret

ConvertToDOSStr endp

; Вывод строки в консоль

; ebx - Указатель на буфер

; ret - eax - Количество действительно выведенных переменных

WriteToConsole proc

    ; Получаем длину строки. Длина строки будет в регистре eax

    push ebx

    call lstrlenA@4

    push 0                      ; Резервный параметр

    push offset tmp\_local\_var   ; Адрес переменной куда будут записано количество фактически выведенных значений

    push eax                    ; Размер выводимой строки

    push ebx                    ; Адрес буфера

    push dout                   ; Дескриптор консоли для вывода

    call WriteConsoleA@20

    mov eax, tmp\_local\_var ; Поместить количество выведенных символов в eax

    ret

WriteToConsole endp

; Функция чтения данных из консоли.

; eax - Указатель на буфер куда читаем

; ebx - Максимальный размер буфера - его длина

; ret - В случае успеха в переменной eax будет хранится значение фактически прочитанных символов c учетом перевода каретки и строки

ReadFromConsole proc

    push 0                    ; Резервный параметр

    push offset tmp\_local\_var ; Адрес переменной, в которую будет помещено еоличество действительно прочитанных символов

    push ebx                  ; Длина буфер

    push eax                  ; Указатель на буфер

    push din                  ; Дескриптор ввода

    call ReadConsoleA@20      ; Вызво операции чтения

    mov eax, tmp\_local\_var ; Положить фактически прочитанное количество символов в eax

    ret

ReadFromConsole endp

; Конвертация строки в число

; esi - указатель на буфер

; ecx - длина строки без учета символов перевода строки и возврата каретки он же счетчик цикла

; edi - Здесь будет храниться наш результат вычисления

; ret - В случаем успеха в переменной edx будет лежать значение числа

StringToInt proc

    ; Проверка длины строки, если она 4 <

    cmp ecx, 4

    jl incorrect\_input\_data\_error

    mov tmp\_local\_var , 1 ; Коэффициент перевода числа для разрядности

    add esi, ecx

    dec esi ; Нужно на 1 сдвинуть вниз

    for\_start:

        mov al, [esi] ; Получить символ с конца строки

        ; Обработка знака минус

        cmp al, '-' ; Провераем равны ли знаки

        ; Если равно, то переход на метку обработки знака минус

        je set\_minus\_bit

        ; Валидация строки

        cmp al , '0' ; Сравнение символа с '0'

        ; Если меньше переход на метку ошибки

        jl not\_a\_number\_char\_error

        cmp al, '9' ; Сравнение символа с '9'

        ; Если больше то переход на метку ошибки

        jg not\_a\_number\_char\_error

        ; Обработка символа

        sub al, '0'   ; Перевод из строки в число

        imul tmp\_local\_var ; Передвинул число на нужный разряд

        jo overflow\_reg\_error ; Проверка на переполнение регистра, если число > INT\_MAX

        add [edi], eax ; Прибавили число к результату

        mov ebx, eax ; Сохранаем последнее прибавлаемое число, необходимо для контроля записи числа, (отсутсвие нуля в входном формате )

        ; Проверка флага переполнения регистра. Регистр OV = 1

        jo overflow\_reg\_error ; Если при сложении произошло переполнение то так же ошибка

        dec esi ; Двинули указатель на буфер

        dec ecx ; уменьшили значение переменной счетчика

        jecxz for\_end ;  Если все закаончилось

        mov eax , tmp\_local\_var ; Загружаем переменную для увеличения базы

        mul mul\_coef                    ; Сдвиг для десятичного разряда

        mov tmp\_local\_var, eax  ; Сохранили сдвиг по разрядам

        ;jo overflow\_reg\_error ; Если при сложении произошло переполнение то так же ошибка

        xor eax, eax ; Сбросили регистр в ноль

        jmp for\_start ; Переход на начало цикла

    for\_end:

        ; Проверка полученного числа. Число должно быть больше или равно 1000

        mov eax, 1000

        cmp [edi], eax

        jl not\_a\_number\_char\_error

        cmp ebx, 0 ; Сравнение последнего введенного числа и нуля

        je not\_a\_number\_char\_error

        ret

    ; Обработка знака минус у числа

    set\_minus\_bit:

        cmp ebx, 0 ; Сравнение последнего введенного числа и нуля

        je not\_a\_number\_char\_error

        mov eax, [edi]

        or eax, 80000000h ; Бит знака устанавливаем

        mov [edi], eax

        ; ecx == 1

        cmp ecx, 1

        ; Если это не так, то это ошибка и у нас не корректная строка

        jne not\_a\_number\_char\_error

        ; Полученное число должно быть  -1000 <

        mov eax, 800003E8h ;  800003E8h = -1000

        cmp [edi], eax

        jl not\_a\_number\_char\_error

        ret

    ; Метка для обработки события некорректно веденного символа или формата числа

    not\_a\_number\_char\_error:

        mov eax, offset str\_not\_correct\_num\_char

        call ConvertToDOSStr

        mov ebx, offset str\_not\_correct\_num\_char

        call WriteToConsole

        push -2

        call ExitProcess@4

    ; Метка обработки переполнения регистра

    overflow\_reg\_error:

        mov eax, offset str\_overflow\_error\_msg

        call ConvertToDOSStr

        mov ebx, offset str\_overflow\_error\_msg

        call WriteToConsole

        push -2

        call ExitProcess@4

    ; Метка некорректно введенного числа

    incorrect\_input\_data\_error:

        mov eax, offset str\_input\_str\_short

        call ConvertToDOSStr

        mov ebx, offset str\_input\_str\_short

        call WriteToConsole

        push -2

        call ExitProcess@4

StringToInt endp

; Коныертация числа в 16 - ый формат и преобразование его в строку

; Вывести результат в консоль в 16 формате

; число лежит в переменной result

IntTo16Str proc

    mov ecx , 0; Счетчик цикла

    mov ebp, -1 ; В случае если в числе есть знак минуса то увеличим его на 1

    ; Проверка знака.

    mov eax, result

    and eax, 80000000h ; Получить бит знака

    and result, 7FFFFFFFh ; Снять бит знака

    mov ebx , result ; Загрузили в переменную значение переменной для вывода

    cmp eax, 80000000h

    je SetMinusSign

    jmp continue

    SetMinusSign:

        mov outbuf[0] , '-'

        inc ebp

        jmp continue

    continue:

        mov ecx, 8 ; Количество итераций, которое нужно провести

    ; В цикле проходим по всем числам и цыклически получаем их

    for\_start:

        mov eax, ebx ; Загрузили переменную result в регистр

        and eax, 0000000Fh ; маска 0000|1111

        ; Получаем из массива символ

        shr ebx, 4 ; сдвиг на 4 бита впарво

        ; А вот костыль по другому не работает

        mov al, table\_char[eax] ; Поместить по индексу код символа

        mov outbuf[ecx + ebp], al

        dec ecx ; i--

        jecxz for\_end ; Если ecx = 0 конец цикла

        jmp for\_start ; На начало цикла

    for\_end:

        ret

IntTo16Str endp

MAIN PROC

; Базовая настройка

call OpenRead  ;Открываем дескриптор чтения

call OpenWrite ;Открываем дескриптор записи в консоль

mov eax , offset strn; Значение второго операнда перемещается в первый, offset заберает адрес

call ConvertToDOSStr

; Ввод первого числа

; Вызов функции для вывода строки приглошения ввести число

mov ebx, offset strn

call WriteToConsole

; Операция ввода строки

mov eax, offset inbuf

mov ebx, 15     ; Размер буфера

call ReadFromConsole ; В регистре eax - фактический размер прочитанной строки с учетом 2 символов

; Указатель на буфер

mov esi, offset inbuf

; Размер массива, ровно все цифры + знак

mov ecx, eax

sub ecx, 2

mov edi, offset num1 ; Указатель на область памяти где будет хранится результат преобразоания

; Конверация строки в число

call StringToInt

mov eax, num1

; Ввод второго числа

; Вызов функции для вывода строки приглошения ввести число

mov ebx, offset strn

call WriteToConsole

; Операция ввода строки

mov eax, offset inbuf

mov ebx, 15     ; Размер буфера

call ReadFromConsole ; В регистре eax - фактический размер прочитанной строки с учетом 2 символов

; Указатель на буфер

mov esi, offset inbuf

; Размер массива, ровно все цифры + знак

mov ecx, eax

sub ecx, 2

mov edi, offset num2 ; Указатель на область памяти где будет хранится результат преобразоания

; Конверация строки в число

call StringToInt

; Умножение чисел

mov eax, num1

and eax, 80000000h ; Получить бит знака

and num1, 7FFFFFFFh ; Снять бит знака

mov ecx, eax ; Сохрянить информацию о знаке

mov eax, num2

and eax, 80000000h ; Получить бит знака

and num2, 7FFFFFFFh ; Снять бит знака

xor ecx, eax ; Оставить знак или нет

mov eax, num1

imul num2 ; Знаковое умножение, нужно, что бы поймать установку бита переполнения

; Контроль переполнения регистра

jo mul\_overflow\_reg\_error

; Востановить бит знака если это необходимо

or eax , ecx

; Сохранаем результат

mov result, eax

call IntTo16Str

; Вывод результата в 16-ом формате

mov eax, offset outbuf

call ConvertToDOSStr

mov ebx, offset outbuf

call WriteToConsole

; return 0;

push 0

call ExitProcess@4

; В случае переполнения в OV=1

mul\_overflow\_reg\_error:

    mov eax, offset str\_overflow\_error\_msg\_mul

    call ConvertToDOSStr

    mov ebx, offset str\_overflow\_error\_msg\_mul

    call WriteToConsole

    push -2

    call ExitProcess@4

MAIN ENDP

END MAIN

## Тесты к программе

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Входные данные** | | **Результат** | | **Комментарий** |
| 1 число (10) | 2 число (10) | Результат (10) | Результат (16) |  |
| 3459 | 5480 | 18955320 | 01213C38 | Тривиальная проверка на правильность |
| 46340 | 46341 | 2147441940 | 7FFF5D14 | Краевая ситуация |
| -46341 | 46340 | -2147441940 | -7FFF5D14 | Проверка работы с отрицательными числами |
| -1234 | -1234 | 1522756 | 00173C44 | Проверка корректной работы при подаче 2-х отрицательных чисел. |
| 0001 |  |  |  | Некорректная входная последовательность. Программа выведет сообщение:  “Некоректный символ и/или запись числа. Допустимые символы: [0-9] и знак минуса” |
| F9340 |  |  |  | Некорректная входная последовательность. Программа выведет сообщение:  “Некоректный символ и/или запись числа. Допустимые символы: [0-9] и знак минуса” |
| 123 |  |  |  | Некорректная входная последовательность. Программа выведет сообщение:  “Введенное число слишком короткое. Длина должна быть 4 символа и более” |
| 21474836478 |  |  |  | Переполнение регистра при считывании числа. Программа выведет:  “Переполнение регистра при конвертации строки в число” |
| 46345 | 46348 |  |  | Переполнение регистра при выполнении операции умножения. Программа выведет:  “Переполнение регистра при конвертации строки в число” |