|  |  |  |
| --- | --- | --- |
|  | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования «Новосибирский государственный технический университет» | | |
|  | | |
| Кафедра теоретической и прикладной информатики | | |
| Лабораторная работа № 1 | | |
| по дисциплине «Языки программирования и методы трансляции» | | |
| **Основные элементы языка ассемблера** | | |
|  | | |
|  | Бригада | Голубь Андрей |
|  | Буданцев Дмитрий |
| Группа ПМ-13 |  |
| Вариант 3 |  |
|  |  |
|  |  |
| Преподаватель | Дворецкая Виктория Константиновна |
|  |  |
| Новосибирск,2024 | | |

1. **Постановка задачи**

Написать программу, используя язык Ассемблера, которая:

* Складывает два числа (не менее 4-х знаков каждое число).
* Два числа вводятся с консоли и записываются в десятичной системе счисления.
* Результат работы программы выводится в консоль в шестнадцатеричной системе счисления.
* Числа выбираются, таким образом, чтобы при выполнении арифметических операций, регистры не переполнялись.
* При выводе результата не использовать wsprintfA

1. **Алгоритм**
   * + 1. Перекодировать все строковые константы с помощью функции CharToOem
       2. Получим необходимые дескрипторы ввода-вывода и сохраним их в переменные DIN и DOUT соответственно
       3. Получим два числа с помощью функции ReadConsole. Каждое число проверим подходит ли оно условию с помощью подпрограммы CHECKNUM, если условие не выполняется выходим из программы с кодом 1, иначе перекодируем два числа из строчного формата в числовой с помощью подпрограммы CONVERTCHARTOINT.
       4. Складываем два числа
       5. Переводим результат шага 4 к шестнадцатеричной системе счисления с помощью подпрограммы CONVERT10TO16.
       6. Выводим результат шага 5.
       7. Завершаем программу с кодом 0.
2. **Результаты тестирования программы**

|  |  |  |
| --- | --- | --- |
| **Входные данные** | **Выходные данные** | **Цель теста** |
| 9555924  45587 | 9281E7 | Проверка корректности работы, простой тест |
| -559844  6844566 | 5FE5B2 | Использование отрицательных чисел |
| 598759  -8875465 | -7E4AE2 | Отрицательный рельтат |
| -587  874654 | ERROR | Использование чисел не подходящие под условие |
| 7  -854756 | ERROR |
| 89998  7 | ERROR |
| 89998  -48 | ERROR |

1. **Текст программы**

.386

.MODEL FLAT, STDCALL

OPTION CASEMAP: NONE

; 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; функция выхода из программы

EXTERN  lstrlenA@4: PROC; функция определения длины строки

.DATA; сегмент данных

STRN1 db "Введите первое число: ",13,10,0; выводимая строка, в конце добавлены

; управляющие символы: 13 – возврат каретки, 10 – переход на новую

; строку, 0 – конец строки; с использованием директивы DB

; резервируется массив байтов

STRN2 db "Введите второе число: ", 13, 10, 0;

DERR db "ERROR: Число введено не верно", 13, 10, 0;

DIN DD ?; дескриптор ввода; директива DD резервирует память объемом

; 32 бита (4 байта), знак «?» используется для неинициализированных данных

DOUT DD ?; дескриптор вывода

BUF  DB 200 dup (?); буфер для вводимых/выводимых строк длиной 200 байтов

LENS DD ?; переменная для количества выведенных символов

VAL SDWORD ?; переменная хранящая значение первого числа

.CODE; сегмент кода

CONVERTCHARTOINT PROC

; Подпрограмма берёт число из BUF, размера LENS и сохраняет в регистр EAX

; Проверяем число является ли отрицательным

MOV ESI, OFFSET BUF

MOV ECX, LENS

XOR EBX, EBX

MOV EAX, 0

MOV BL, [ESI];

CMP BL, '-'

JNE NO\_Otr

JE Otr

NO\_Otr: ; Если не отрицательно

    CONVERT:

        MOV EDX, 10

        MOV BL, [ESI];

        SUB BL, '0';

        MUL EDX;

        ADD EAX, EBX

        INC ESI;

    LOOP CONVERT;

    jmp end\_proces

Otr: ; Если отрицательно

    INC ESI;

    CONVERT\_1:

        MOV EDX, 10

        MOV BL, [ESI];

        SUB BL, '0';

        IMUL EDX;

        ADD EAX, EBX

        INC ESI;

    LOOP CONVERT\_1;

    MOV EDX, -1

    IMUL EDX

end\_proces:

    ret

CONVERTCHARTOINT ENDP

CONVERT10TO16 PROC

; Переводит результат в шестнадцатеричный формат

; Первый параметр число которое переводится

MOV ESI, OFFSET BUF

PUSH OFFSET BUF

CALL lstrlenA@4

MOV ECX, EAX

CLEAR\_LIST:

    MOV EBX, 0

    MOV [ESI], EBX

    INC ESI

LOOP CLEAR\_LIST

MOV ESI, OFFSET BUF

MOV EAX, VAL

CMP EAX, 0

JL Be

JA next

JE next

Be:

    MOV EBX, '-'

    MOV [ESI], EBX

    INC ESI

    MOV EDX, -1

    IMUL EDX

next:

MOV EBX, 16

MOV ECX, EAX

MOV EBP, 0

CONVERT\_2:

    XOR EDX, EDX

    DIV EBX

    CMP EDX, 9

    JA large

    JE ok\_num

    JB ok\_num

    large:

        SUB EDX, 10

        ADD EDX, 'A'

        PUSH EDX

        jmp end\_if

    ok\_num:

        ADD EDX, '0'

        PUSH EDX

    end\_if:

    INC EBP

    MOV ECX, EAX

    INC ECX

LOOP CONVERT\_2

MOV ECX, EBP

TO\_BUF:

    POP EAX

    MOV [ESI], EAX

    INC ESI

LOOP TO\_BUF

ret

CONVERT10TO16 ENDP

CHECKNUM PROC

; Проверить является ли число подходящим под условие

; Условие: число имеет минимум 4 символа

; Нужно: не пустой BUF, LENS

MOV ESI, OFFSET BUF

MOV BL, [ESI];

CMP BL, '-'

JE Minus

JNE Not\_minus

Minus:

    SUB LENS, 1

Not\_minus:

CMP LENS, 4

JB Error

JE Not\_err

JA Not\_err

Error:

    PUSH OFFSET DERR

    CALL lstrlenA@4; длина в EAX

    PUSH 0; в стек помещается 5-й параметр

    PUSH OFFSET LENS; 4-й параметр

    PUSH EAX; 3-й параметр

    PUSH OFFSET DERR; 2-й параметр

    PUSH DOUT; 1-й параметр

    CALL WriteConsoleA@20

    PUSH 1

    CALL ExitProcess@4

Not\_err:

    ret

CHECKNUM ENDP

MAIN PROC; начало описания процедуры с именем MAIN

MOV  EAX, OFFSET STRN1; командой MOV  значение второго операнда

; перемещается в первый, OFFSET – операция, возвращающая адрес

PUSH EAX; параметры функции помещаются в стек командой PUSH

PUSH EAX

CALL CharToOemA@8;

MOV  EAX, OFFSET STRN2

PUSH EAX

PUSH EAX

CALL CharToOemA@8; вызов функции

MOV  EAX, OFFSET DERR

PUSH EAX

PUSH EAX

CALL CharToOemA@8; вызов функции

; получим дескриптор ввода

PUSH -10

CALL GetStdHandle@4

MOV DIN, EAX    ; переместить результат из регистра EAX

; в ячейку памяти с именем DIN

; получим дескриптор вывода

PUSH -11

CALL GetStdHandle@4

MOV DOUT, EAX

; определим длину строк STRN

PUSH OFFSET STRN1; в стек помещается адрес строки

CALL lstrlenA@4; длина в EAX

PUSH 0; в стек помещается 5-й параметр

PUSH OFFSET LENS; 4-й параметр

PUSH EAX; 3-й параметр

PUSH OFFSET STRN1; 2-й параметр

PUSH DOUT; 1-й параметр

CALL WriteConsoleA@20

PUSH 0; в стек помещается 5-й параметр

PUSH OFFSET LENS; 4-й параметр

PUSH 200; 3-й параметр

PUSH OFFSET BUF; 2-й параметр

PUSH DIN; 1-й параметр

CALL ReadConsoleA@20 ; обратите внимание: LENS больше числа введенных

; символов на два, дополнительно введенные символы: 13 – возврат каретки и

; 10 – переход на новую строку

; вывод полученной строки

SUB LENS, 2;

CALL CHECKNUM;

CALL CONVERTCHARTOINT;

MOV VAL, EAX

PUSH OFFSET STRN2; в стек помещается адрес строки

CALL lstrlenA@4; длина в EAX

PUSH 0; в стек помещается 5-й параметр

PUSH OFFSET LENS; 4-й параметр

PUSH EAX; 3-й параметр

PUSH OFFSET STRN2; 2-й параметр

PUSH DOUT; 1-й параметр

CALL WriteConsoleA@20

PUSH 0; в стек помещается 5-й параметр

PUSH OFFSET LENS; 4-й параметр

PUSH 200; 3-й параметр

PUSH OFFSET BUF; 2-й параметр

PUSH DIN; 1-й параметр

CALL ReadConsoleA@20 ; обратите внимание: LENS больше числа введенных

; символов на два, дополнительно введенные символы: 13 – возврат каретки и

; 10 – переход на новую строку

; вывод полученной строки

SUB LENS, 2;

CALL CHECKNUM;

CALL CONVERTCHARTOINT;

ADD VAL, EAX

CALL CONVERT10TO16

PUSH OFFSET BUF

CALL lstrlenA@4;

PUSH 0

PUSH OFFSET LENS

PUSH EAX

PUSH OFFSET BUF

PUSH DOUT

CALL WriteConsoleA@20

PUSH 0; параметр: код выхода

CALL ExitProcess@4

MAIN ENDP; завершение описания модуля с указанием первой выполняемой процедуры

END MAIN;