|  |  |  |
| --- | --- | --- |
|  | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования FPMI_ngtu_neti_rgb_polya«Новосибирский государственный технический университет» | | |
|  | | |
| Кафедра прикладной математики | | |
| Практическое задание № 1 | | |
| по дисциплине «Языки программирования и методы трансляции» | | |
| **ОСНОВНЫЕ ЭЛЕМЕНТЫ ЯЗЫКА АССЕМБЛЕРА** | | |
|  | | |
|  | Группа ПМ-04 | Плешкова Дарья |
| Бригада 11 | Сальников дмитрий |
|  |  |
|  |  |
|  |  |
|  |  |
| Преподаватель | сивак мария алексеевна |
|  | петров роман владимирович |
| Новосибирск, 2022 | | |

**1. Задание**

Умножение чисел, ввод в двоичной системе счисления (не менее 16-ти знаков каждое число), вывод в десятичной системе счисления.

**2. Алгоритм**

* Получить дескрипторы ввода и вывода
* Вывести сообщение для ввода числа
* Считать первое число
* Преобразовать строку в число и записать в оперативную память
* Считать второе число
* Преобразовать строку в число и записать в оперативную память
* Умножножить два числа, результат записать в регистр
* Преобразуем число в строку, чтобы было как число в 10-й системе счисления
* Вывести получившуюся строку

**3. Тесты**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Первое число (в двоичной СС)** | **Второе число (в двоичной СС)** | **Результат (в десятичной СС)** | **Комментарий** |
| **1.** | 1 | 1 | 1 | Простой тест (1 \* 1) |
| **2.** | 111 | 111 | 49 | Тест с одинаковыми входными данными (7 \* 7) |
| **3.** | 111 | 110 | 42 | Тест с разными входными данными (7 \* 6) |
| **4.** | 0 | 111 | 0 | Тест с нулевым входным значением (0 \* 7) |
| **5.** | 10(х16) | 10(х15) | Ошибка | Тест на переполнение  (216 \* 215 = 231) |
| **6.** | 10 | 1(х30) | 2147483646 | Тест на максимально возможное число  (2 \* 230 = 231 - 2) |
| **7** | 2 | - | Ошибка | Первое число не подходит под 2ую систему счисления |
| **8** | 0 | 5 | Ошибка | Второе число не подходит под 2ую систему счисления |
| **9** | Ю | - | Ошибка | Первый символ не является числом, а так же числом в 2ой системе счисления |
| **10** | 1 | Ю | Ошибка | Второй символ не является числом, а так же числом в 2ой системе счисления |
| **11** | 11ю | - | Ошибка | Строка содержит символ не относящийся к 2ой системе счисления |
| **12** | 1 | 11ю | Ошибка | Строка содержит символ не относящийся к 2ой системе счисления |

**4. Используемые программные средства языка Ассемблер**

**Push** – помещает содержимое регистра, ячейки памяти или константу размером в слово или двойное слово на вершину стека.

**Pop** – снимает слово или двойное слово с вершины стека и помещает его в ячейку памяти или регистр.

**Mov** – команда, которая пересылает байт, слово или двойное слово между регистром и ячейкой памяти или между двумя регистрами.

**Xor** – команда обнуляющая регистр, выполняющая побитную операцию «исключающее или»

**Inc** – команда добавляет 1 к содержимому регистра или ячейки памяти.

**Call** – вызвает процедуру.

**Add** – команда складывает содержимое операнда-источника и операнда-приемнка и помещает результат в операнд-приемник

**Sub** – команда вычитает операнд-источник из операнда-приемника и возвращает результат в операнд-приемник

**Imul** – команда умножает числа числа со знаком.

**Div** – команда выполняет деление чисел без знака,

**Cmp** – вычитает исходный операнд из операнда получателя данных и, в зависимости от полученного результата, устанавливает флаги состояния процессора.

**Jb** – команда условия перехода, где приемник меньше источника

**Jmp** – команда безусловного перехода, заставляет процессор извлечь новую команду не из следующей ячейки памяти, а из ячейки, связанной с операндом команды.

**Jae** – команда условия перехода, где приемник больше источника или равен ему**5. Текст программы**

.386 ; директива, предписывающая Ассемблеру использовать набор операций для процессора 80386.

.MODEL FLAT, STDCALL ; плоская модель, вызов процедур по соглашению в Windows x32, далее - процедуры:

EXTRN GetStdHandle@4:PROC ; стандартный дескриптор ввода-вывода

EXTRN WriteConsoleA@20:PROC ; вывод текста

EXTRN CharToOemA@8:PROC ; перекодировка

EXTRN ReadConsoleA@20:PROC ; ввод

EXTRN ExitProcess@4:PROC ; выход

EXTRN lstrlenA@4:PROC ; определение длинны строки

.DATA ; сегмент данных

DOUT DD ? ; дескриптор вывода

DIN DD ? ; дескриптор ввода

STRN DB "Введите первое число: ",13,10,0 ; выводимая строка

STRN2 DB "Введите второе число: ",13,10,0 ; выводимая строка

STRNRES DB "Результат умножения: ", 13,10,0 ; выводимая строка

BUF DB 35 dup (?) ; достаточный буфер для вводимых/выводимых строк

LENS DD ? ; количество выведенных символов

;два вводимых числа

NUMEROS\_A DD ? ; первое число

NUMEROS\_B DD ? ; второе число

; переменные для хранения констант - оснований систем счислени

ZWEY DD 2 ; присвоение основание системы счисления для 2

TEN DD 10 ; присвоение основание системы счисления для 10

.CODE ; сегмент кода

MAIN PROC

; метка точки входа

START:

; перекодируем строку STRN

PUSH OFFSET STRN ; параметры функции помещаются в стек командой PUSH, OFFSET — операция, возвращающая адрес (смещения)

PUSH OFFSET STRN

CALL CharToOemA@8

; получим дескриптор ввода

PUSH -10 ; константа ввода

CALL GetStdHandle@4

MOV DIN, EAX ; перемещение результата из регистра EAX в ячейку памяти с именем DIN (для дескриптора ввода)

; получим дескриптор вывода

PUSH -11 ; константа вывода

CALL GetStdHandle@4

MOV DOUT, EAX ; перемещение результата из регистра EAX в ячейку памяти с именем DOUT (для дескриптора вывода)

Go1:

; Первое число!

; вывод строки

PUSH OFFSET STRN ; в стек помещается указатель на строку

CALL lstrlenA@4 ; длина в EAX

; вызов функции WriteConsoleA для вывода строки STRN

PUSH 0 ; 5-й параметр (Конец аргументов)

PUSH OFFSET LENS ; 4-й параметр (Место куда поместить кол-во выведенных символов)

PUSH EAX ; 3-й параметр (Длина выводимой строки)

PUSH OFFSET STRN ; 2-й параметр (Адрес выводимой строки)

PUSH DOUT ; 1-й параметр (Дескриптор вывода)

CALL WriteConsoleA@20

; ввод строки

PUSH 0 ; 5-й параметр (Конец аргументов)

PUSH OFFSET LENS ; 4-й параметр (Длина введенных символов с учетом 2 служебных)

PUSH 35 ; 3-й параметр (Длина строки)

PUSH OFFSET BUF ; 2-й параметр (Куда записать введенную строку)

PUSH DIN ; 1-й параметр (Дескриптор ввода)

CALL ReadConsoleA@20

; обработка 1-го числа (на данный момент число виде строки в буфере)

SUB LENS, 2 ; вычитаем служебные символы (Определяем фактическую длину строки)

MOV ECX, LENS ; счетчик цикла - количество необработанных символов

MOV ESI, OFFSET BUF ; начало строки хранится в переменной BUF

XOR BX, BX ; обнулить регистр BX командой XOR, выполняющей побитно операцию «исключающее или»

XOR AX, AX ; обнулить регистр AX

CONVERT\_1: ; метка начала тела цикла

MOV BL, [ESI] ; поместить символ из введенной строки в регистр BL, используя косвенную адресацию

CMP BL, '2'

JB Next1

JMP Go1

Next1:

CMP BL, '0'

JAE Next2

JB Go1

Next2:

SUB BL, '0' ; вычесть из введенного символа код нуля

MUL ZWEY ; умножить значение AX на 2, результат – в AX

ADD AX, BX ; добавить к полученному в AX числу новую цифру

INC ESI ; перейти на следующий символ строки

LOOP CONVERT\_1 ; перейти на следующую итерацию цикла

MOV NUMEROS\_A, EAX ; отправим полученное число в память

; Второе число!

; перекодируем строку STRN2

PUSH OFFSET STRN2 ; параметры функции помещаются в стек командой PUSH, OFFSET — операция, возвращающая адрес (смещения)

PUSH OFFSET STRN2

CALL CharToOemA@8

Go2:

; вывод строки

PUSH OFFSET STRN2 ; в стек помещается указатель на строку

CALL lstrlenA@4 ; длина в EAX

; вызов функции WriteConsoleA для вывода строки STRN

PUSH 0 ; 5-й параметр (Конец аргументов)

PUSH OFFSET LENS ; 4-й параметр (Место куда поместить кол-во выведенных символов)

PUSH EAX ; 3-й параметр (Длина выводимой строки)

PUSH OFFSET STRN2 ; 2-й параметр (Адрес выводимой строки)

PUSH DOUT ; 1-й параметр (Дескриптор вывода)

CALL WriteConsoleA@20

; ввод строки

PUSH 0 ; 5-й параметр (Конец аргументов)

PUSH OFFSET LENS ; 4-й параметр (Длина введенных символов с учетом 2 служебных)

PUSH 35 ; 3-й параметр (Длина строки)

PUSH OFFSET BUF ; 2-й параметр (Куда записать введенную строку)

PUSH DIN ; 1-й параметр (Дескриптор ввода)

CALL ReadConsoleA@20

; обработка 2-го числа (на данный момент число виде строки в буфере)

SUB LENS, 2 ; вычитаем служебные символы (Определяем фактическую длину строки)

MOV ECX, LENS ; счетчик цикла - количество необработанных символов

MOV ESI, OFFSET BUF ; начало строки хранится в переменной BUF

XOR BX, BX ; обнулить регистр BX командой XOR, выполняющей побитно операцию «исключающее или»

XOR AX, AX ; обнулить регистр AX

CONVERT\_2: ; метка начала тела цикла

MOV BL, [ESI] ; поместить символ из введенной строки в регистр BL, используя косвенную адресацию

CMP BL, '2'

JB Next3

JMP Go2

Next3:

CMP BL, '0'

JAE Next4

JB Go2

Next4:

SUB BL, '0' ; вычесть из введенного символа код нуля

MUL ZWEY ; умножить значение AX на 2, результат – в AX

ADD AX, BX ; добавить к полученному в AX числу новую цифру

INC ESI ; перейти на следующий символ строки

LOOP CONVERT\_2 ; перейти на следующую итерацию цикла

MOV NUMEROS\_B, EAX ; отправим второе число

; умножение чисел

MOV EAX, NUMEROS\_A ; отправим первое число в регистр

MOV EBX, NUMEROS\_B ; отправим второе число в регистр

IMUL EAX, EBX ; умножим, результат в EAX, отправлять обратно в память не будем, результат в EAX понадобится

; вывод результата

CDQ ; приведём тип к 64-х битному (EAX распространяется на EDX)

XOR EDI, EDI ; обнуление командой XOR, выполняющей побитно операцию «исключающее или»

MOV ESI, OFFSET BUF ; начало строки хранится в переменной BUF

.WHILE EAX>=TEN ; пока число > 10

DIV TEN ; делим число на 10, результат целая часть в EAX, остаток в EDX

ADD EDX, '0' ; добавить код нуля

PUSH EDX ; помещаем данные в стек для инвертирования

ADD EDI, 1 ; прибавим единицу

XOR EDX, EDX ; очистка реестра EDX

.ENDW ; конец цикла

; обработаем последнее число

ADD EAX, '0' ; добавить кода нуля

PUSH EAX ; в стек

ADD EDI, 1 ; прибавить единицу

; теперь инвертируем строку

MOV ECX, EDI ; Помещаем количество цифр в регистр ECX

CONVERT\_3: ; начало цикла

POP [ESI] ; обращаемся к байту на который указывает ESI и перезаписываем его на значение лежащее в стеке

INC ESI ; увеличить ESI на 1

LOOP CONVERT\_3 ; конец цикла

; перекодируем строку STRNRES

PUSH OFFSET STRNRES ; параметры функции помещаются в стек командой PUSH, OFFSET — операция, возвращающая адрес (смещения)

PUSH OFFSET STRNRES

CALL CharToOemA@8

; вывод строки

PUSH OFFSET STRNRES ; В стек помещается указатель на строку

CALL lstrlenA@4 ; Определение длины строки STRNRES (результат записан в EAX)

; Вызов функции WriteConsoleA для вывода строки STRNRES

PUSH 0 ; 5-й параметр (Конец аргументов)

PUSH OFFSET LENS ; 4-й параметр (Место куда поместить кол-во выведенных символов)

PUSH EAX ; 3-й параметр (Длина выводимой строки)

PUSH OFFSET STRNRES ; 2-й параметр (Адрес выводимой строки)

PUSH DOUT ; 1-й параметр (Дескриптор вывода)

CALL WriteConsoleA@20

; в стек помещается указатель на строку

PUSH OFFSET BUF ; В стек помещается указатель на строку

CALL lstrlenA@4

PUSH 0 ; 5-й параметр (Конец аргументов)

PUSH OFFSET LENS ; 4-й параметр (Место куда поместить кол-во выведенных символов)

PUSH EAX ; 3-й параметр (Длина выводимой строки)

PUSH OFFSET BUF ; 2-й параметр (Адрес выводимой строки)

PUSH DOUT ; 1-й параметр (Дескриптор вывода)

CALL WriteConsoleA@20

; выход из программы

PUSH 0 ; код выхода

CALL ExitProcess@4

MAIN ENDP

END MAIN