|  |  |  |
| --- | --- | --- |
| Министерство науки и высшего образования  Российской Федерации | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования | | |
| «Новосибирский государственный технический университет» | | |
|  | | |
| Теоретической и прикладной математики | | |
|  | | |
| Лабораторная работа № 1 | | |
| по дисциплине «Низкоуровневое программирование» | | |
|  | | |
| **ОСНОВНЫЕ ЭЛЕМЕНТЫ ЯЗЫКА АССЕМБЛЕРА** | | |
|  | | |
|  | Факультет: | ПМИ |
| Группа: | ПМИ-02 |
| Вариант: | 7 |
| Студент: | Сидоров Даниил, |
|  | Дюков Богдан |
| Преподаватель: | Еланцева И. Л., |
|  | Сивак М. А.  . |
|
|  |  |
|  | | |
| Новосибирск | | |
| 2022 | | |

1. **Цель работы**

Изучить и приобрести практические навыки работы с основными командами языка Ассемблера, функциями ввода-вывода, регистрами и символьными данными.

1. **Содержание работы**

* Изучить основные команды языка Ассемблера, функции операционной системы, осуществляющие ввод-вывод, работу символьными данными
* Разработать программу, реализующую умножение чисел, ввод в восьмеричной системе счисления (не менее 3-х знаков каждое число) и вывод в десятичной системе счисления;
* Отладить программу, убедиться в правильности ее работы на тестовых примерах.

1. **Текст программы**

**.386 ; это ассемблерная директива, предписывающая Ассемблеру использовать**

**; набор операций для процессора 80386**

**.model flat, stdcall ; плоская модель памяти. Эта модель памяти используется в операционной системе Windows.**

**; stdcall — используемое соглашение о вызовах процедур**

**; прототипы внешних процедур описываются директивой extrn,**

**; после знака @ указывается общая длина передаваемых параметров,**

**; после двоеточия указывается дистанция «ближняя» – near**

**extrn  GetStdHandle@4:near ; функция получения дескриптора (функция возвращает дескриптор в EAX)**

**extrn  WriteConsoleA@20:near**

**extrn  CharToOemA@8:near ; функция перекодировки строковых констант в DOS**

**extrn  ReadConsoleA@20:near**

**extrn  ExitProcess@4:near ; функция выхода из программы**

**extrn  lstrlenA@4:near ; функция определения длины строки**

**; сегмент данных**

**.data**

**dout dd ? ; дескриптор вывода (директива dd резервирует память объемом 32 бита (4 байта)), знак вопроса обозначает неинициализированные данные**

**din dd ? ; дескриптор ввода**

**strn1 db "Введите первое число: ",13,10,0 ; первая выводимая строка,**

**strn2 db "Введите второе число: ",13,10,0 ; вторая выводимая строка,**

**strn3 db "Результат: ",13,10,0 ; третья выводимая строка,**

**; управляющие символы: 13 – возврат каретки, 10 – переход на новую строку, 0 – конец строки**

**; (с использованием директивы db резервируется массив байтов)**

**buf  db 200 dup (?); буфер для вводимых/выводимых строк**

**lens dd ? ; количество выведенных символов**

**; два вводимых числа**

**numa dd ?**

**numb dd ?**

**; переменные для учета минусов в введенных числах**

**minus1 dd ?**

**minus2 dd ?**

**; сегмент кода**

**.code**

**start: ; метка точки входа (завершается двоеточием)**

**; перекодируем строку strn1**

**push offset strn1 ; параметры функции помещаются в стек командой**

**; offset – операция, возвращающая смещение**

**push offset strn1**

**call CharToOemA@8 ; вызов функции**

**; перекодируем строку strn2**

**push offset strn2 ; параметры функции помещаются в стек командой**

**; offset – операция, возвращающая смещение**

**push offset strn2**

**call CharToOemA@8 ; вызов функции**

**; перекодируем строку strn3**

**push offset strn3 ; параметры функции помещаются в стек командой**

**; offset – операция, возвращающая смещение**

**push offset strn3**

**call CharToOemA@8 ; вызов функции**

**; получим дескриптор ввода**

**push -10**

**call GetStdHandle@4**

**mov din, eax ; переместим результат из регистра eax в ячейку памяти с именем din**

**; получим дескриптор вывода**

**push -11**

**call GetStdHandle@4**

**mov dout, eax**

**; вывод строки**

**try1:push offset strn1 ; в стек помещаем указатель на строку**

**call lstrlenA@4 ; длина в eax**

**push 0 ; в стек помещаем 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**

**; ввод строки**

**push 0 ; в стек помещаем 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**

**; обработка 1 строки**

**; помещаем в регистры ecx и esi число итераций цикла и начало введенной строки соответственно**

**xor eax, eax**

**sub lens, 2**

**mov ecx, lens**

**mov esi, offset buf ; начало строки храним в переменной buf**

**; запоминаем и убираем минус**

**mov bl, [esi]**

**.if bl=='-' ;**

**mov minus1, 1**

**inc esi**

**dec ecx**

**.endif**

**xor ebx, ebx ; очищаем ebx**

**xor eax,  eax ; очищаем eax**

**convert: ; метка начала цикла**

**xor edx, edx ; очищаем edx**

**mov dl, 8 ; на это число будем умножать, делаем в цикле т.к. при умножении dl затирается**

**mov bl, [esi] ; помещаем символ из введенной строки в bl**

**; Проверка на корректность ввода**

**.if (bl>='0') && (bl<='7')**

**sub bl, '0' ; вычитаем из введенного символа код нуля**

**mul edx ; умножаем старое значение eax на 8, результат – в eax**

**add eax, ebx ; добавляем к полученному числу новое значение**

**inc esi ; переходим на следующую строку**

**.else**

**mov minus1, 0**

**jmp try1**

**.endif**

**loop convert ; новая итерация цикла**

**mov numa, eax ; сохраняем введенное число в переменную numa**

**; вывод строки**

**try2:push offset strn2 ; в стек помещаем указатель на строку**

**call lstrlenA@4**

**push 0 ; в стек помещаем 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**

**; ввод строки**

**push 0 ; в стек помещаем 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**

**; обработка 2 строки**

**xor eax, eax**

**sub lens, 2**

**mov ecx, lens**

**mov esi, offset buf ; начало строки храним в переменной buf**

**; запоминаем и убираем минус**

**mov bl, [esi]**

**.if bl=='-' ;**

**mov minus2, 1**

**inc esi ;**

**dec ecx**

**.endif**

**xor ebx, ebx ; очищаем ebx**

**xor eax, eax ; очищаем eax**

**convertb: ; метка начала цикла**

**xor edx,edx ; очищаем edx**

**mov dl, 8 ; на это число будем умножать, делаем в цикле т.к. при умножении dx затирается**

**mov bl, [esi] ; помещаем символ из введенной строки в bl**

**; проверка на корректность ввода**

**.if (bl>='0') && (bl<='7')**

**sub bl, '0' ; вычитаем из введенного символа код нуля**

**mul dx ; умножаем старое значение bx на 10, результат – в ax**

**add eax, ebx ; добавляем к полученному числу новое значение**

**inc esi ; переходим на следующую строку**

**.else**

**mov minus2,0**

**jmp try2**

**.endif**

**loop convertb ; новая итерация цикла**

**mov numb, eax**

**; умножаем числа, результат будет помещен в eax**

**mov eax, numa**

**mov ebx, numb**

**mul ebx**

**; проверка на переполнение**

**jo fail**

**; преобразование результата**

**xor edi, edi**

**xor edx, edx**

**mov esi, offset buf ; начало строки храним в переменной buf**

**xor ecx, ecx**

**mov ecx, 10**

**mov ebx, eax ; в ebx храним полное число**

**.while ebx>=10 ; пока число >= 10**

**div ecx**

**add edx, '0'**

**push edx ; кладем данные в стек, для инвертирования**

**add edi, 1**

**xor edx, edx**

**xor ebx, ebx**

**mov ebx, eax**

**.endw**

**add eax, '0'**

**push eax**

**add edi, 1**

**; добавление минуса, если необходимо**

**mov ebx, minus1**

**mov ecx, minus2**

**.if (ecx!=ebx) && (numa != 0) && (numb != 0)**

**mov ebx,'-'**

**push ebx**

**add edi, 1**

**.endif**

**; теперь инвертируем строку**

**mov ecx, edi**

**convertc:**

**pop [esi]**

**inc esi**

**loop convertc**

**push offset strn3 ; в стек помещаем указатель на строку**

**call lstrlenA@4 ; длина в eax**

**push 0 ; в стек помещаем 5-й параметр**

**push offset lens ; 4-й параметр**

**push eax ; 3-й параметр**

**push offset strn3 ; 2-й параметр**

**push dout ; 1-й параметр**

**call WriteConsoleA@20**

**; выводим результат**

**push offset buf ; в стек помещаем указатель на строку**

**call lstrlenA@4**

**push 0 ; в стек помещаем 5-й параметр**

**push offset lens ; 4-й параметр**

**push eax ; 3-й параметр**

**push offset buf ; 2-й параметр**

**push dout ; 1-й параметр**

**call WriteConsoleA@20**

**; выход из программы**

**fail: push 0 ; параметр: код выхода**

**call ExitProcess@4**

**end start**

1. **Тесты**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Первое число** | **Второе число** | **Результат** | **Комментарий** |
| 1 | 7 | 15 | 91 | Простейший тест |
| 2 | 356 | 674 | 105672 | Умножение чисел, состоящих из 3 знаков |
| 3 | 177777 | 177777 | 4294836225 | Умножение чисел большой размерности |
| 4 | 200000 | 200000 | 0 | Тест на переполнение |
| 5 | -5671 | 545 | -1071357 | Умножение положительного и отрицательного числа |
| 6 | -1 | -1 | 1 | Умножение двух отрицательных числа |
| 7 | -1 | 0 | 0 | 0 без знака |
| 8 | 0033 | 044 | 972 | С нулями в начале числа |
| 9 | Вводим не числовое значение или число, не входящее в 8-ю систему счисления | | Ввод данных будет продолжаться пока не будут введены корректные данные | Проверка некорректного ввода |

1. **Вывод**

Контрольные вопросы проработаны.