Новосибирский государственный технический университет

Кафедра ТПИ

Низкоуровневое программирование

Лабораторная работа №1

**Основные элементы языка Assembler**

Выполнили: Преподаватели:

Яшков Иван Сивак Мария Алексеевна

Гавриленко Андрей

Группа:

ПМИ-13

Бригада 1

Новосибирск 2023

**1. Задание**

Сложение чисел, ввод в восьмеричной системе счисления (не менее 3-х знаков каждое число), вывод в десятичной системе счисления.

**2. Алгоритм решения:**

Задача была разбита на несколько логических подзадач:

1. Пользовательский ввод двух чисел

2. Преобразование строк в числа

3. Подсчет результата сложения

4. Преобразование результата в строку

5. Получение результата пользователем

Используемые переменные:

INPUT – Строка-приглашение для ввода пользователем числа

OUTPUT – Строка предваряющая вывод результата

STRN1 – Буфер для первого числа

STRN2 – Буфер для второго числа

RESULT – Буфер для результата

DIN – Переменная для запоминания дескриптора ввода

DOUT – Переменная для запоминания дескриптора вывода

NUMBER1 – Первое число после преобразование из строки

NUMBER2 – Второе число после преобразования из строки

LEN – Вспомогательная переменная – количество символов

SIGN – Служит для запоминания знака каждого из чисел

MINUS – Хранит символ “-“

В программе используются функции:

GetStdHandle@4

WriteConsoleA@20

CharToOemA@8

ReadConsoleA@20

ExitProcess@4

lstrlenA@4

1. Пользовательский ввод двух чисел:

Получаем дескрипторы для ввода и вывода в консоль с помощью функции GetStdHandle@4, записываем их в переменные. Последовательно предлагаем пользователю ввести два числа в восьмеричной системе счисления, используя команду WriteConsoleA@20, сохраняем введенные строки с помощью функции ReadConsoleA@20.

2. Преобразование строк в числа:

Сначала первый символ каждой строки сравнивается с символом “-“ командой CMPSB, для этого в регистр ESI помещается адрес строки, а в регистр EDI адрес переменной MINUS. Если такой символ присутствует, значение регистра ESI увеличивается на 1, переменная, хранящая длину строки, уменьшается на 1. Далее, вне зависимости от первого символа, значение переменной, хранящей длину строки, уменьшается еще на 2, так как в строку входят символы \r\n. После этого в регистр ECX помещаем значение этой переменной. Далее в цикле считывается каждый символ строки, вычитаем из него код “0” для преобразования в число и прибавляем к результату. На следующей итерации цикла результат умножаем на основание системы счисления (8) и прибавляем новое число.

3. Подсчет результата сложения:

На предыдущем шаге мы получили 2 числа. Теперь складываем их.

4. Преобразование результата в строку:

Теперь нужно перевести сумму, полученную на предыдущем шаге в строку для последующего вывода в консоль. Сначала проверим, отрицательно полученное число или нет. Для этого сравним его с нулем, если оно меньше или равно, то на первую позицию результирующей строки поместим минус. Далее изменим знак числа: сделаем это с помощью комбинации команд NOT и INC. На этом этапе мы в любом случае имеем положительное число.

Теперь нужно использовать цикл. Но перед этим зададим значение регистра ECX равным 0, а само число поместим в регистр EAX. В цикле задаем EDX равным 0, далее используем команду IDIV, деля число на основание нужной системы счисления (10). Получаем остаток в регистре EDX и частное в регистре EAX. Заносим значение регистра EDX в стек командой PUSH. Инкрементируем значение регистра ECX. Цикл продолжается, пока значение регистра EAX больше 0.

После этого в следующем цикле (LOOP) достаем значение из стека командой POP, прибавляем к нему код “0” и записываем на место следующего символа в результирующей строке.

После этого цикла дописываем в конце строки завершающий символ “0”.

5. Получение результата пользователем:

Сначала используем команду WriteConsoleA@20 для вывода предшествующей строки, после этого с помощью такой команду выводим результирующую строку.

**3. Код 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

INPUT DB "Введите число в восьмеричной системе счисления: ", 13,10,0 ;Строка - приглашение ввода

OUTPUT DB "Результат: ", 13,10,0 ;Строка - перед выводом результата

STRN1 DB 50 dup(?) ;Строка для ввода первого числа

STRN2 DB 50 dup(?) ;Строка для ввода второго числа

RESULT DB 50 dup(?) ;Строка, в которую запишем результат

DIN DD ? ;Дескриптор ввода

DOUT DD ? ;Дескриптор вывода

NUMBER1 DD ? ;Первое число и результат

NUMBER2 DD ? ;Второе число

LEN DD ? ;Вспомогательная переменная-количество символов

SIGN DD 1 ;Знак числа, может измениться на -1

MINUS DB "-" ;Минус

.CODE; сегмент кода

MAIN PROC; начало описания процедуры с именем MAIN

;Получим дескрипторы ввода и вывода

PUSH -10

CALL GetStdHandle@4

MOV DIN, EAX

PUSH -11

CALL GetStdHandle@4

MOV DOUT, EAX

;Преобразуем строки в DOS-кодировку

MOV EAX, OFFSET INPUT

PUSH EAX

PUSH EAX

CALL CharToOemA@8

MOV EAX, OFFSET OUTPUT

PUSH EAX

PUSH EAX

CALL CharToOemA@8

;Вывод приглашения ввода для первого числа

PUSH OFFSET INPUT

CALL lstrlenA@4

PUSH 0

PUSH OFFSET LEN

PUSH EAX

PUSH OFFSET INPUT

PUSH DOUT

CALL WriteConsoleA@20

;Ввод первого числа

PUSH 0

PUSH OFFSET LEN

PUSH 50

PUSH OFFSET STRN1

PUSH DIN

CALL ReadConsoleA@20

;Перевод первой строки в число

;Проверка знака

MOV ESI, OFFSET STRN1

MOV EDI, OFFSET MINUS

CMPSB

JNE @I1

SUB LEN, 1

MOV SIGN, -1

INC ESI

@I1: DEC ESI

SUB LEN, 2 ;Отнимаем два символа - \r\n

MOV DI, 8

MOV ECX, LEN

XOR AX, AX

XOR BX, BX

WHEEL:

MOV BL, [ESI]

SUB BL, '0'

MUL DI

ADD AX, BX

INC ESI

LOOP WHEEL

IMUL SIGN

MOV NUMBER1, EAX

;Вывод приглашения ввода для второго числа

PUSH OFFSET INPUT

CALL lstrlenA@4

PUSH 0

PUSH OFFSET LEN

PUSH EAX

PUSH OFFSET INPUT

PUSH DOUT

CALL WriteConsoleA@20

;Ввод второго числа

PUSH 0

PUSH OFFSET LEN

PUSH 50

PUSH OFFSET STRN2

PUSH DIN

CALL ReadConsoleA@20

;Перевод второй строки в число

;Проверка знака

MOV SIGN, 1

MOV ESI, OFFSET STRN2

MOV EDI, OFFSET MINUS

CMPSB

JNE @I2

SUB LEN, 1

MOV SIGN, -1

INC ESI

@I2: DEC ESI

SUB LEN, 2 ;Отнимаем два символа - \r\n

MOV DI, 8

MOV ECX, LEN

XOR AX, AX

XOR BX, BX

ROLL:

MOV BL, [ESI]

SUB BL, '0'

MUL DI

ADD AX, BX

INC ESI

LOOP ROLL

IMUL SIGN

MOV NUMBER2, EAX

;CMP NUMBER2, 0 Тест на отрицательное число

;JGE @T1

; MOV NUMBER2, 0

; MOV NUMBER2, 0

;@T1:

; Сложение двух чисел

ADD NUMBER1, EAX

; Перевод суммы в строку

MOV EAX, NUMBER1

MOV EDI, OFFSET RESULT

MOV ECX, 0

CMP EAX, 0

MOV EBX, 10

JGE @SPIN

MOV DL, MINUS

MOV [EDI], DL

INC EDI

NOT EAX

INC EAX

@SPIN:

MOV EDX, 0

IDIV EBX

PUSH EDX

INC ECX

CMP EAX, 0

JG @SPIN

TWIRL:

POP EDX

ADD Dl, 30h

MOV [EDI], DL

INC EDI

LOOP TWIRL

MOV DL, 0

MOV [EDI], DL

;INC EDI

;MOV EAX, EDI

;SUB EAX, OFFSET RESULT

; Вывод строки, предшествующей результату

PUSH OFFSET OUTPUT

CALL lstrlenA@4

PUSH 0

PUSH OFFSET LEN

PUSH EAX

PUSH OFFSET OUTPUT

PUSH DOUT

CALL WriteConsoleA@20

; Вывод результата

PUSH OFFSET RESULT

CALL lstrlenA@4

PUSH 0

PUSH OFFSET LEN

PUSH EAX

PUSH OFFSET RESULT

PUSH DOUT

CALL WriteConsoleA@20

PUSH 0

CALL ExitProcess@4

MAIN ENDP; завершение описания процедуры с именем MAIN

END MAIN; завершение описания модуля с указанием первой выполняемой процедуры