**Лабораторная работа № 1**

Тема: Технология работы с программами на языке Ассемблера. Элементарные операции.

1. Цель работы

Ознакомиться с технологией написания и отладки программ на языке

ассемблера. Ознакомиться со структурой программы и директивами языка.

Ознакомиться с операциями пересылки данных, передачи управления и прерываниями ввода-вывода.

2. Краткие теоретические сведения

Ассемблер — это программа, которая переводит текст с языка, понятного человеку, в язык, понятный процессору, то есть говорят, что она переводит язык ассемблера в машинный код.

Система программирования «*Turbo Assembler»*от фирмы Borland в основном предназначена для создания 16-битных программ для DOS и процессоров семейства Intel x86. Система включает в себя следующие основные компоненты

[5]:

1)Компилятор языка ассемблер – *tasm.exe*– программа, предназначенная для компиляции файла написанного на языке ассемблера (*.asm*) в объектные модули

(*.obj*).

2)Компоновщик (linker) – *tlink.exe*– программа, предназначенная для сборки исполняемого файла из объектных модулей и библиотек. Основные два формата исполнимых файлов в DOS — *.com*и *.exe*. Файлы типа *.com*содержат только скомпилированный код без какой-либо дополнительной информации о программе. Весь код, данные и стек такой программы располагаются в одном сегменте и не могут превышать 64 килобайта. Файлы типа *.exe*содержат заголовок, в котором

описывается размер файла, требуемый объем памяти, список команд в программе, использующих абсолютные адреса, которые зависят от расположения

программы в памяти, и т.д. *Exe*-файл может иметь любой размер.

3) Отладчик (debugger) – *td.exe*– программа, предназначенная для отладки,

созданных исполняемых файлов.

Все указанные выше программы получают опции через командную строку, все

доступные функции можно получить, запустив программу без параметров.

Кроме этого, Borland предоставляет интегрированную среду «Turbo Assembler»

– *ta.exe*– программу, объединяющую текстовый редактор и указанные выше компоненты. Эта программа похожа по своим функциональным возможностям на

среду «Turbo C».

Как указано выше, исполнимые файлы могут быть двух форматов. Рассмотрим

два примера на языке ассемблера с использованием упрощенных директив

описания сегментов.

Пример программы типа *.com*:

;hello1.asm

;Выводит на экран сообщение "Hello World!" и завершается

|  |  |  |  |
| --- | --- | --- | --- |
| .model | tiny | ; | модель памяти, используемая для СОМ |
| .code |  | ; | начало сегмента кода |
| org | 100h | ; | начальное значение счетчика - 100h |
| start: mov | ah,9 | ; | номер функции DOS - в АН |
| mov | dx,offset message ; | | адрес строки - в DX |
| int | 21h | ; | вызов системной функции DOS |
| ret |  | ; | завершение СОМ-программы |
| message db | "Hello World!",0Dh,0Ah,'$' ; строка для вывода | | |
| end | start | ; | конец программы |

Для формирования *.com*программы используйте команды:

tasm.exe hello1.asm hello1.obj

tlink.exe /t hello1.obj

Пример программы типа *.exe*:

;hello2.asm

;Выводит на экран сообщение "Hello World!" и завершается

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| .model | small | ; модель памяти, используемая для ЕХЕ | | | |
| .stack | 100h | ; сегмент стека размером в 256 байт | | | |
| .code |  | ; | | сегмент кода | |
| start: mov | ax,DGROUP | ; | | адрес сегмента данных | |
| mov | ds,ax | | ; помещается в DS | |
| mov | dx,offset message | |  | |
| mov | ah,9 | |  | |
| int | 21h | | ; функция DOS "вывод строки" | |
| mov | ax,4C00h | |  | |
| int | 21h | | ; функция DOS "завершить программу" | |
| .data |  | | ; сегмент данных | |
| message db | "Hello World!",0Dh,0Ah,'$' | | | |
| end | start | |  | |

Для формирования *.exe*программы используйте команды:

tasm.exe hello2.asm hello2.obj

tlink.exe hello2.obj

Для выполнения лабораторной работы необходимо изучить архитектуру процессора Intel 8086 (реальный режим). Эти данные можно взять из [1, 7, 9, 14].

Также необходимо изучить директивы, элементарные операции ассемблера и прерывания ввода-вывода данных. К ним относятся:

1) *Директивы определения данных*– указывают ассемблеру, что в

соответствующем месте программы располагается переменная, определяют тип

переменной (байт, слово, вещественное число и т.д.), задают ее начальное значение и ставят в соответствие переменной метку, которая будет

использоваться для обращения к этим данным. Определения данных

записываются в общем виде следующим образом:

|  |  |
| --- | --- |
| имя\_переменной D\* | значение |

где D\* может определять: DB – байт, DW – слово (2 байта), DD - двойное слово (4 байта), DF — байт, DQ — 8 байт, DT — 10 байт. Поле значения может содержать одно или несколько чисел, строк символов (взятых в одиночные или двойные кавычки), операторов ? и DUP, разделенных запятыми. Все

установленные таким образом данные окажутся в выходном файле, а имя переменной будет соответствовать адресу первого из указанных значений. Если вместо точного значения указан знак ?, переменная считается

неинициализированной и ее значение на момент запуска программы может

оказаться любым. Если нужно заполнить участок памяти повторяющимися

данными, используется специальный оператор DUP, имеющий формат

счетчик DUP (значение)

2) *Команды пересылки данных и способы адресации.*Базовой командой

пересылки данных является команда MOV. Эта команда копирует содержимое

источника в приемник, источник не изменяется. Команда MOV действует аналогично операторам присваивания из языков высокого уровня, то есть команда MOV AX,BX присваивает регистру AX значение из регистра BX. В качестве

источника для MOV могут использоваться: число (непосредственный операнд), регистр общего назначения, сегментный регистр или переменная (то есть

операнд, находящийся в памяти). В качестве приемника — регистр общего назначения, сегментный регистр (кроме CS) или переменная. Оба операнда должны быть одного и того же размера — байт, слово или двойное слово. Нельзя

выполнять пересылку данных с помощью MOV из одной переменной в другую, из одного сегментного регистра в другой и нельзя помещать в сегментный регистр непосредственный операнд — эти операции выполняют двумя командами MOV. Также в данной работе можно использовать стек – это специальным образом организованный участок памяти, используемый для временного хранения переменных, для передачи параметров вызываемым подпрограммам и для сохранения адреса возврата при вызове процедур и прерываний. Данные можно записывать и считывать только с вершины стека. Таким образом, если записать в стек числа 1, 2, 3, то при чтении они будут получаться в обратном порядке — 3, 2, 1. Для работы со стеком используются команды PUSH (поместить данные в стек) и POP (считать данные из стека). Способы адресации определяют доступ к данным. Часто используемые способы адресации:

|  |  |
| --- | --- |
| -- регистровая | – MOV AX,BX |
| -- непосредственная | – MOV AX,2 |
| -- прямая | – MOV AX,ES:0001 |
| -- косвенная | – MOV AX,[BX] |
| -- по базе со сдвигом | – MOV AX,[BX+2] |
| -- по базе с индексированием | – MOV AX,[BX+SI+2] |

Более подробно ознакомиться со способами адресации можно в [2, 3, 6 - 13]. 3) *Команды передачи управления*– служат для организации вычислительного

процесса. Предлагается использовать следующие команды этой группы:

|  |  |
| --- | --- |
| -- безусловный переход | – JMP |
| -- условный переход | – Jcc (где cc – условие перехода, обычно |

используется после команды CMP)

|  |  |
| --- | --- |
| -- цикл | – LOOP |
| -- прерывание | – INT |

Более подробно ознакомиться с командами передачи управления можно в [2, 3, 6

-13].

4)*Прерывания ввода-вывода*– функции BIOS или DOS, предоставляющие

сервис по работе с аппаратурой ПЭВМ. Для ввода данных с клавиатуры предлагается использовать одну из ниже приведенных функций DOS:

-- *Функция DOS 01h*— Считать символ из STDIN с эхом, ожиданием и

проверкой на Ctrl-Break

|  |  |
| --- | --- |
| Ввод: | АН = 01h |
| Вывод: | AL = ASCII-код символа или 0. Если AL = 0, второй вызов этой |

функции возвратит в AL расширенный ASCII-код символа Особенности: При чтении с помощью этой функции введенный символ

автоматически немедленно отображается на экране (посылается в устройство STDOUT — так что его можно перенаправить в файл). При нажатии Ctrl-C или Ctrl-Break выполняется команда INT 23h.

Если нажата клавиша, не соответствующая какому-нибудь символу

(стрелки, функциональные клавиши Ins, Del и т.д.), то в AL возвращается 0 и функцию надо вызвать еще один раз, чтобы получить расширенный ASCII-код.

-- *Функция DOS 06h*— Считать символ из STDIN без эха, без ожидания и без проверки на Ctrl-Break

|  |  |
| --- | --- |
| Ввод: | АН = 06h |
|  | DL = 0FFh |
| Вывод: | ZF = 1, если не была нажата клавиша, и AL = 00 |
|  | ZF = 0, если клавиша была нажата. В этом случае AL = код символа |

Особенности: Вариант функции 01h

-- *Функция DOS 07h*— Считать символ из STDIN без эха, с ожиданием и без проверки на Ctrl-Break

|  |  |
| --- | --- |
| Ввод: | АН = 07h |
| Вывод: | AL = код символа |

Особенности: Вариант функции 01h

-- *Функция DOS 08h*— Считать символ из STDIN без эха, с ожиданием и

проверкой на Ctrl-Break

|  |  |
| --- | --- |
| Ввод: | АН = 08h |
| Вывод: | AL = код символа |

Особенности: Вариант функции 01h

-- *Функция DOS 0Ah*— Считать строку символов из STDIN в буфер

|  |  |
| --- | --- |
| Ввод: | АН = 0Ah |
|  | DS:DX = адрес буфера |
| Вывод: | Буфер содержит введенную строку |

Особенности: Для вызова этой функции надо подготовить буфер, первый байт

которого содержит максимальное число символов для ввода (1 – 254), а содержимое, если оно задано, может использоваться как подсказка для ввода. При наборе строки обрабатываются клавиши

Esc, F3, F5, BS, Ctrl-C/Ctrl-Break и т.д., как при наборе команд DOS

(то есть Esc начинает ввод сначала, F3 восстанавливает подсказку для ввода, F5 запоминает текущую строку как подсказку, Backspace стирает предыдущий символ). После нажатия клавиши Enter строка (включая последний символ CR (0Dh)) записывается в буфер,

начиная с третьего байта. Во второй байт записывается длина

реально введенной строки без учета последнего CR.

Для вывода данных на консоль предлагается использовать одну из ниже

приведенных функций DOS:

-- *Функция DOS 02h*— Записать символ в STDOUT с проверкой на Ctrl-Break

|  |  |
| --- | --- |
| Ввод: | АН = 02h |
|  | DL = ASCII-код символа |
| Вывод: | AL = код последнего записанного символа (равен DL, кроме случая, |
|  | когда DL = 09h (табуляция), тогда в AL возвращается 20h). |

Особенности: Эта функция при выводе на экран обрабатывает некоторые

управляющие символы — вывод символа BEL (07h) приводит к звуковому сигналу, символ BS (08h) приводит к движению курсора влево на одну позицию, символ НТ (09h) заменяется на несколько

пробелов, символ LF (0Ah) опускает курсор на одну позицию вниз, и CR (0Dh) приводит к переходу на начало текущей строки. Если в

|  |  |
| --- | --- |
|  | ходе работы этой функции была нажата комбинация клавиш Ctrl- |
|  | Break, вызывается прерывание 23h, которое по умолчанию |
|  | осуществляет выход из программы. |
| -- *Функция DOS 06h*— Записать символ в STDOUT без проверки на Ctrl-Break | |
| Ввод: | АН = 06h |
|  | DL = ASCII-код символа (кроме FFh) |
| Вывод: | AL = код записанного символа (копия DL) |
| Особенности: Эта функция не обрабатывает управляющие символы (CR, LF, HT и | |
|  | BS выполняют свои функции при выводе на экран, но сохраняются |
|  | при перенаправлении вывода в файл) и не проверяет нажатие Ctrl- |
|  | Break. |
| - *Функция DOS 09h*— Записать строку в STDOUT с проверкой на Ctrl-Break | |
| Ввод: | АН = 09h |
|  | DS:DX = адрес строки, заканчивающейся символом $ (24h) |
| Вывод: | AL = 24h (код последнего символа) |

Особенности: Действие этой функции полностью аналогично действию функции

02h, но выводится не один символ, а целая строка.

- *Функция DOS 40h*— Записать в файл или устройство

|  |  |
| --- | --- |
| Ввод: | АН = 40h |
|  | ВХ = 1 для STDOUT или 2 для STDERR |
|  | DS:DX = адрес начала строки |
|  | СХ = длина строки |
| Вывод: | CF = 0, |
|  | АХ = число записанных байт |

Особенности: Эта функция предназначена для записи в файл, но, если в регистр

ВХ поместить число 1, функция 40h будет выводить данные на STDOUT, а если ВХ = 2 — на устройство STDERR. STDERR всегда выводит данные на экран и не перенаправляется в файлы. На этой функции основаны используемые в С функции стандартного вывода — фактически функция С fputs() просто вызывает это

прерывание, помещая свой первый аргумент в ВХ, адрес строки (второй аргумент) — в DS:DX и длину — в СХ.

# 3. Порядок выполнения работы

Написать программу на ассемблере, выполняющую операции над строкой

символов согласно заданиям, приведенным ниже. Общие требования:

-- выделить буфер для хранения 200 символов; -- строку символов ввести с клавиатуры;

-- ввод строки символов может быть завершен клавишей «Enter» или по

заполнению буфера;

-- дополнительный буфер для хранения промежуточных результатов обработки

строки не вводить;

-- константы задать с помощью директивы EQU;

-- старт программы, ввод – вывод данных и обработку ошибок оформлять

выводом в консоли поясняющих строк.

## 4. Задания к лабораторной работе

1) Отсортировать символы в строке по значению ASCII кода символа.

2) Выполнить реверс слов строки.

3) Выполнить реверс символов слов строки.

4) Отсортировать слова в строке (варианты: по длине слова, по значению ASCII кодов символов, по алфавиту).

5) Удалить заданное (варианты: стоящее перед заданным, стоящее после заданного) слово в строке.

6) Заменить заданное слово (подстроку) в строке на другую заданную подстроку.

7) Вставить в строке перед заданным словом другое заданное слово.

8) Удалить во всех строках текста слова, содержащие заданный набор букв.

9) Заменить во всех строках текста слова, содержащие заданный набор букв,

на другое заданное слово.

10) Вставить во всех строках текста слово "number" перед словами,

состоящими из цифр.

11 Удалить во всех строках текста слова являющиеся числами.

12) Заменить во всех строках текста слова являющиеся числами на слово

"<number>".