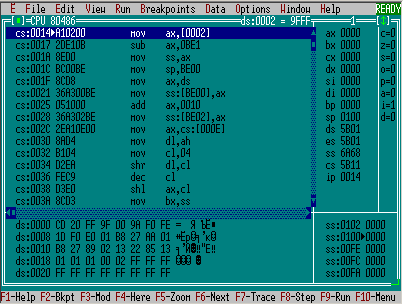
Основы работы в отладчике Turbo Debugger.

Отладчик Turbo Debugger (TD) является достаточно мощным средством для отладки программ. Он позволяет изучать и отлаживать функционирования программ, написанных не только с использованием Ассемблера, а и написанных языками высокого уровня – C и Pascal. Он позволяет выполнять программы по одной команде, устанавливать контрольные точки, следить за дампом памяти, флагами и регистрами процессора и сопроцессора. Turbo Debugger является незаменимым инструментом при написании программ на Ассемблере.

Проще всего вызвать TD, указав в командной строке имя exe-файла для исследования:

*td lab1.exe*

После загрузки TD перед пользователем появится окно (CPU Window):



Окно CPU состоит из пяти частей:

1. Код программы.

Текущая команда выделена синим цветом. Кроме ассемблерной команды в этом окне отображается ее код и адрес (пара сегмент-смещение). Вместо адреса сегмента, как правило, выводится регистр CS. Это значит, что адрес начала сегмента с кодом, который содержит текущую команду, находится в этом регистре.

На одну из команд указывает стрелка, которая показывает, какая команда будет выполнена следующей.

В предыдущем примере текущая инструкция находится в памяти по адресу CS:0014, она будет выполнена следующей, ее код - A10200, соответствующая ассемблерная команда – “mov ax,[0002]”.

Если одним из операндов команды является ячейка памяти (независимо от способа адресации), то в оглавлении окна CPU будет отображено ее значение (например, “ds:0002 = 9FFF”).

1. Значения регистров (находятся правее от кода программы).

В этой области приводятся значения всех программно-доступных регистров процессора. Причем если во время выполнения предыдущей команды значение регистра изменилось, для удобства он выделяется белым цветом. Можно убедится, что в приведенном примере адрес следующей команды (CS:0014) задается парою регистров CS:IP (CS=5B11, IP=0014).

1. Значения флагов (правее значений регистров).

Для удобства восприятия регистр флагов не выводится правее с другими регистрами, а отображается отдельно побитово.

1. Дамп памяти (ниже кода программ).

В этой области отображается некоторая область памяти, значения ячеек которой можно изменять при желании. Слева отображается адрес (пара сегмент-смещение) первого слева байта. Потом – 8 байт, дальше – буквенный эквивалент каждого байта (соответственно ASCII-кода).

1. Стек.

В этой области отображается верхушка стека процессора.

Переход между областями окна CPU осуществляется с помощью клавиши <Tab> или мышки.

При нажатии правой кнопки мыши или клавиши <Alt+F10> появляется локальное меню активной области CPU, с помощью которого можно выполнять некоторые операции. Краткий перечень этих локальных меню приведено ниже:

Локальное меню кода программы (не полное)

┌─────────────────┐

│ Goto... │ -осуществляет переход по нужному адресу

│ Origin │ -осуществляет переход на активную команду

│ Follow │ -разрешает проследить, куда будет осущетсвлен следующий переход

│ Search... │ -осуществляет поиск заданной последовательности байтов

├─────────────────┤

│ New cs:ip │ -делает текущую команду активной

│ Assemble... │ -изменяет текущую команду

└─────────────────┘

Локальное меню значений регистров

┌───────────────────────┐

│ Increment │ -инкрементирует текущий регистр

│ Decrement │ -декрементирует текущий регистр

│ Zero │ -обнуляет текущий регистр

│ Change... │ -изменяет значения текущего регистра на заданное

│ Registers 32-bit No │ -задает тип отображаемых регистров (16 или 32 разряда)

└───────────────────────┘

Локальное меню флагов

┌────────┐

│ Toggle │ -инверсия флага

└────────┘

Локальное меню дампа памяти (не полное)

┌─────────────┐

│ Goto... │ -осуществляет переход по заданному адресу памяти

│ Search... │ -осуществляет поиск заданной последовательности байтов

│ Next │ -переход на следующую метку поиска

│ Change... │ -изменить значение байта

│ Previous │ -переход на предыдущую метку поиска

└─────────────┘

Для активизации главного меню TD следует нажать <F10>.

Рассмотрим основные команды меню Turbo Debugger.

Меню File

Open... –открывает новый файл для отладки

Change dir... –изменить текущий каталог

Get info... –отобразить информацию о программе, которая отлаживается

DOS shell -временный выход в DOS

───────────────────

Quit Alt-X -выход

Меню View

Stack -отображение стека

File... –просмотреть файл

CPU -открыть окно CPU

Dump -разрешает открыть дамп памяти

Registers -просмотр регистров

При выполнении команды Dump открывается окно, аналогичное нижней части окна CPU, в котором отображается содержимое некоторой области памяти.

Меню Run

Run F9 -выполняет программу

Go to cursor F4 -выполняет программу до той команды, на которую указывает курсор

Trace into F7 -выполняет текущую команду

Step over F8 -выполняет текущую команду

Execute to... Alt-F9 -выполняет программу до указанного адреса

Until return Alt-F8 -выполняет программу до возврата из процедуры

───────────────────────────

Program reset Ctrl-F2 -останавливает выполнение программы

Разница между Trace into и Step over в том, что в случае выполнения процедуры первая начнет отлаживать ее, а другая выполнит как одну обыкновенную инструкцию. После остановки програмы она переходит в такое же состояние как и после начальной загрузки программы для отладки.

Подготовка программы к выполнению.

Процесс подготовки и отладки программы языка ассемблера включает следующие этапы: подготовка начального текста, трансляция, компоновка и отладка.

Подготовка текста программы осуществляется с помощью любого текстового редактора. Единственное требование при этом – файл должен быть сохранен в ASCII-кодах, иначе ваша программа не будет воспринята Ассемблером.

Трансляция текста программы состоит в преобразовании строк начальной программы в коды машинных команд. Она выполняется с помощью транслятора из языка ассемблера. После трансляции программы создается объектный файл (с расширением obj).

Трансляция программы осуществляется с помощью команды :

# Tasm [<имя файла.asm>]

Некоторые коррективы программы Tasm приведены ниже:

/l -создает листинг программы

/la -создает расширенный листинг программы

/w0,/w1,/w2 -устанавливает уровень предупреждений (/w0 - нет;/w1,/w2 -есть

/z -вместе с информацией об ошибке выводить строку входного файла

/zi,/zn,/zd -информация для отладки(/zi-полная;/zd-только номера строк;/zn-нет)

Компоновка объектного файла осуществляется с помощью программы-компоновщика (редактора связей). Пример задания команды компоновки приведен ниже:

# Tlink [<имя файла.obj>]

Директивы:

/v -сохранение информации для отладки

/t -создание COM-файла

Исполняемый файл, например lab1.exe, можно отладить, используя команду td lab1.exe или выполнить, если в программе предусмотрен вывод на экран.