* flat assembler 1.73

Руководство пользователя

Томаш Грыштар

# Введение

Эта глава содержит всю важнейшую информацию, которая понадобится вам, чтобы начать использовать flat assembler (FASM). Если у вас уже есть опыт программирования на ассемблере, вам достаточно прочитать лишь первую главу перед использованием этого компилятора.

## Обзор компилятора

Flat assembler – быстрый многопроходный компилятор языка ассемблера. За счет совершения множественных проходов он позволяет использовать выражения до их объявления, разрешает использование условий и циклов. Если в качестве целевой используется архитектура x86, являющаяся для данного ассемблера «родной», то для нее (для каждого конкретного исходного текста) за счет множественных проходов генерируется минимально возможный по размеру машинный код.

Исходные тексты flat assembler представлены на языке самого flat assembler.

Компилятор реализован с полностью идентичной логикой компиляции под различными операционными системами, реализации отличаются только специфичными для системы форматом файла, используемым набором функций для работы с файлами, работы с памятью и работы с пользовательским интерфейсом. В основном это консольные приложения окружающей операционной системы, но для Windows компилятор дополнительно представлен в виде IDE с графическим интерфейсом и встроенным редактором, а также в виде динамически подгружаемой библиотеки. Консольные реализации компилятора имеют имя **fasm** и расширение стандартное для исполняемых файлов операционной системы, реализация в виде динамически подгружаемой библиотеки называется **fasm.dll**, а реализации в виде IDE называются **fasmw.exe** (для графического интерфейса Windows) и **fasmd.exe** (для интерфейса DPMI DOS совместимого).

В компиляторе реализован принцип SSSO (Same source – same output), в связи с этим, один и тот же исходный текст, независимо от всего (от варианта реализации компилятора, от особенностей окружающей операционной системы или от переопределенных настроек самого компилятора), может скомпилироваться в выходной файл только одного и того же содержания.

### Системные требования

Flat assembler – запускается только в окружении архитектуры x86 c 32-разрядной адресацией памяти (т.е. требуется как минимум процессор 80386). Если сосредоточится на реализации под Windows, то консольный вариант требует Win32 совместимую операционную систему (т.е. минимум Windows 3.0), а вариант в виде IDE требуют Win32 GUI совместимую операционную систему (т.е. минимум Windows95). Варианты компилятора, портированые для 64-разрядного окружения, требуют, чтобы 64-разрядная операционная система была способна предоставить все ресурсы программе в пределах 32-разрядных адресов.

#### Требования к файлам исходных текстов

Файл исходного текста представляет (должен представлять) собой обычный текстовый файл, который может быть создан в любом текстовом редакторе, разрывы строк в котором допускаются как в DOS, так и в UNIX стандартах.

#### Рекомендуемые требования к окружению

Применительно к пакету для Windows, исходные тексты программ-примеров поставляемых в этом пакете требуют, чтобы глобальная переменная окружения **INCLUDE** хранила полный путь к папке **include**, которая является частью пакета. Если такая переменная уже существует в вашей системе и содержит пути, используемые другой программой, достаточно добавить к ней новый путь (разные пути разделяются точкой с запятой).

Вместо глобальной переменной окружения можно задать локальную (действующую только для **fasmw.exe**) одноименную переменную в секции Environment файла настроек **fasmw.ini** (автоматически создаваемого при запуске fasmw.exe). Например, если вы распаковали файлы FASM'а в папку **c:\fasmw**, вы должны добавить следующие строки в файл **c:\fasmw\fasmw.ini**:

|  |
| --- |
| [Environment]  Include = c:\fasmw\include |

***Примечание***: В локальной переменной также можно указать несколько путей (разные пути разделяются точкой с запятой).

***Примечание 2***: Если вы не определите переменную окружения **INCLUDE** правильно, вам придется вручную указывать полный путь к заголовочным файлам Win32 в каждой программе, которую вы хотите скомпилировать.

### IDE вариант компилятора: использование

Запустить fasmw.exe можно непосредственно или путем перетаскивания на него другого файла. Если с компилятором связать файлы определенного расширения, также можно будет запускать его по двойному щелчку по файлам этого расширения.

IDE представляет собой окно редактора, и строку меню над ним. В нижней части окна редактора находятся закладки открытых файлов, при щелчке по которым соответствующий файл отображается для редактирования.

Новые файлы можно открыть или через меню **File** команду **Open**, или простым перетаскиванием файла в окно редактора.

По умолчанию компилятор будет обрабатывать файл отображенный в окне редактора, но можно принудить компилятор обрабатывать конкретный файл независимо от того какой файл редактируется в настоящий момент, для этого на соответствующей закладке правым щелчком открыть контекстное меню и выбрать пункт **Assign**. Одновременно к компилятору может быть привязан только один файл.

Когда ваш исходный файл готов, вы можете выполнить компиляцию, выбрав команду **Compile** из меню **Run**. Если компиляция пройдет успешно, компилятор отобразит окно результатов компиляции; иначе он выведет информацию о произошедших ошибках. Окно результатов компиляции содержит информацию о количестве проходов, длительности компиляции и количестве байт записанных в результирующий файл. Оно также содержит текстовое поле называемое Display, в котором отображаются все сообщения от директив display в исходном коде (см. 2.2.3). Сводка ошибок содержит как минимум сообщение об ошибке и текстовое поле **Display**, того же назначения. Если ошибка связана с конкретной строкой исходного кода, сводка содержит также текстовое поле **Instruction**, которое содержит препроцессированную форму инструкции вызвавшей ошибку если ошибка произошла после стадии препроцессора (иначе поле пустое) и список **Source**, который показывает расположение всех строк исходного кода связанных с этой ошибкой, если вы выберете строку из этого списка, она одновременно выбрана в окне редактора (если соответствующий файл не открыт, он будет автоматически загружен).

Команда **Run** также выполняет компиляцию, и в случае успешного её завершения запускает скомпилированную программу в том случае, если она относится к одному из форматов, запускаемых в среде Windows, иначе выводится сообщение о том, что такой тип файла не может быть запущен. Если при компиляции возникают ошибки, выводится сводка по ошибкам, как для команды **Compile**.

Если компилятору не хватает памяти, вы можете увеличить используемый её объём, открыв окно **Compiler Setup** из меню **Options**. В нем вы можете указать объём памяти в килобайтах который компилятор должен использовать и также задать приоритет потока компилятора.

### IDE вариант компилятора: «горячие клавиши»

|  |  |
| --- | --- |
| Перемещение:  Стрелка «←»  Стрелка «→»  Стрелка «↑»  Стрелка «↓»  Ctrl+Стрелка «←»  Ctrl+Стрелка «→»  Home  End  PageUp  PageDown  Ctrl+Home  Ctrl+End  Ctrl+PageUp  Ctrl+PageDown | на 1 символ влево  на 1 символ вправо  на 1 строку вверх  на 1 строку вниз  на 1 слово влево  на 1 слово вправо  на начало строки  в конец строки  на 1 экран вверх  на 1 экран вниз  на первую строку экрана  на последнюю строку экрана  на первую строку программы  на последнюю строку программы |

*Примечание*: При зажатом «Shift» вместо перемещения происходит выделение.

|  |  |
| --- | --- |
| Редактирование:  Insert  Alt+Insert  Delete  Backspace  Ctrl+Backspace  Alt+Backspace  Alt+Shift+Backspace  Ctrl+Y  F6  Операции с блоками:  Ctrl+Insert  Shift+Insert  Ctrl+Delete  Shift+Delete  Ctrl+A  Поиск:  F5  F7  Shift+F7  Ctrl+F7  Компиляция:  F9  Ctrl+F9  Shift+F9  Ctrl+F8  Прочие клавиши:  F2  Shift+F2  F4  Ctrl+N  Ctrl+Tab  Ctrl+Shift+Tab  Alt+[1-9]  Esc  Alt+X  Ctrl+F6  Alt+Стрелка «←»  Alt+Стрелка «→»  Alt+Стрелка «↑»  Alt+Стрелка «↓» | переключение режима вставка/перезапись  переключение режима гор./верт. Выделение  удалить текущий символ  удалить предшествующий символ  удалить предшествующее слово  отмена предыдущей операции (также Ctrl+Z)  возврат отмененной операции (Ctrl+Shift+Z)  удалить текущую строку  дублировать текущую строку  скопировать блок в буфер (также Ctrl+C)  вставить блок из буфера (также Ctrl+V)  удалить блок  вырезать блок в буфер (также Ctrl+C)  выделить все  перейти на заданную строку (также Ctrl+G)  поиск (также Ctrl+F)  поиск следующего (также F3)  замена (также Ctrl+H)  компилировать и запустить  только компилировать  принудить выбранный файл компилятору  компиляция с добавлением отладочной информации  сохранить текущий файл  сохранить файл под новым именем  загрузить файл  создать новый файл  переключиться на следующий файл  переключиться на предыдущий файл  переключиться на файл с заданной вкладки  закрыть текущий файл  закрыть все файлы и выйти из программы  вызвать калькулятор  горизонтальный скрулбар - влево  горизонтальный скрулбар - вправо  вертикальный скрулбар - вверх  вертикальный скрулбар - вниз |

Клавиши специфичные только для IDE fasmw.exe:

|  |  |
| --- | --- |
| F1  Alt+F1 | поиск ключевого слова в справке  вызов контекстной помощи |

### IDE вариант компилятора: опции

В меню **Options** содержится список опций редактора, который может быть включен или выключен в зависимости от текущего состояния редактора. В этом разделе описываются эти опции.

**Secure selection** (безопасное выделение) – будучи включенной, опция снимает выделение перед любым событием печати или удаления символа, таким образом, выделенный блок не затирается, как это бы обычно произошло при отключенной опции. Блочное удаление Ctrl+Delete удаляет выделенный блок независимо от состояния этой опции.

**Automatic brackets** (автоматические скобки) ― будучи включенной, опция при вводе открывающей скобки любого типа, автоматически сразу после нее добавляет такую же закрывающую скобку, а каретку помещает в позицию между этими скобками.

**Automatic indents** (автоматические отступы) ― будучи включенной, опция при создании новой строки сдвигает каретку и текст после нее не на начало новой строки, а так чтобы и каретка и первый не пустой символ новой строки оказались в той же позиции, что и первый не пустой символ строки предшествующей.

**Smart tabulation** (смарт табуляция) ― при табуляции каретка сдвигается вперед на позицию кратную 8, но если опция включена и в пределах этого сдвига в предшествующей строке встречаются не пустой символ, следующий сразу за пустым символом, каретка сдвигается только на позицию этого не пустого символа.

**Optimal fill on saving** (оптимальное сохранение) ― будучи включенной, опция при сохранении файла заменяет все пустые участки не просто пробелами, а оптимальным сочетанием пробелов и табуляций, таким образом, уменьшая размер файла. Независимо от опции пустые символы в конце строк отбрасываются при сохранении.

**Revive dead keys** (Оживление мертвых клавиш) ― будучи включенной, опция отключает в редакторе так называемые мертвые клавиши. Мертвые клавиши (есть не во всех раскладках клавиатуры) – это такие клавиши, которые определяют не полную форму печатного символа, а лишь основной ее символ, диакритический знак этого символа определяет следующая клавиша или (обычно) пробел, если никакого диакритического знака не требуется. Опция может быть полезна если в коде часто приходится использовать основной символ мертвой клавиши и функционала мертвых клавиш не требуется.

**Time scrolling** (прокрутка буфера отмены действий) ― будучи включенной, опция позволяет при зажатой AltGr (или зажатой паре Ctrl+Alt) использовать колесо мыши для прокрутки между состояниями буфера отмены действий.

### Консольный вариант компилятора: использование

Для запуска FASMа из командной строки вам понадобится ввести хотя бы один параметр, указав в нем путь к файлу с исходным текстом. Если необходимо переопределить имя и расширение выходного файла отличное от тех, которые компилятор выберет автоматически, тогда указывается второй параметр содержащий путь к будущему выходному файлу.

Для настройки процесса компиляции в командную строку могут быть добавлены дополнительные параметры. Каждый из таких параметров (не разрывается пробелом) начинается с символа «-» и заканчивается числовым значением:

Параметр -m, переопределяет максимальный размер памяти, который может быть доступен компилятору. В случае DOS реализации значение этой опции ограничено только использованием дополнительной памяти.

Параметр -p, переопределяет максимальное число проходов, за которые ассемблер должен разрешить все неопределенности кода. Значение этой опции может варьироваться в пределах от 0 до 65536, а по умолчанию равно 100.

Опций способных повлиять на содержание выходного файла не предусмотрено.

### Консольный вариант компилятора: сообщения

Вначале показывается краткая информация о названии программы и её версии. Затем в случае успешной компиляции выводится ее краткий итог, включающий информацию о количестве проходов, о продолжительности процесса по времени, о размере созданного выходного файла.

|  |
| --- |
| flat assembler version 1.73 (16384 kilobytes memory)  38 passes, 5.3 seconds, 77824 bytes. |

В случае ошибки во время компиляции, программа выведет на экран сообщение об ошибке. Например, когда компилятор не может найти файл ввода, он покажет следующее сообщение:

|  |
| --- |
| flat assembler version 1.73 (16384 kilobytes memory)  error: source file not found. |

Если ошибка связана с определенной частью кода, будет выведена строка, которая её вызвала. Также, чтобы помочь вам найти эту ошибку, будет показано точное место этой строки в совокупном исходном коде с указанием имени файла и номера строки в нем, например:

|  |
| --- |
| flat assembler version 1.73 (16384 kilobytes memory)  example.asm [3]:  mob     ax,1  error: illegal instruction. |

Это сообщение гласит, что в третьей строке файла «example.asm» находится неразрешенная инструкция.

В случае если ошибку вызвала строка, содержащая макроинструкцию, вместе с ней будет показана строка в определении этой макроинструкции, также с указанием точного места этой строки в совокупном исходном коде:

flat assembler version 1.73 (16384 kilobytes memory)

example.asm [6]:

stoschar 7

example.asm [3] stoschar [1]:

mob     ax,1

error: illegal instruction.

Это сообщение гласит, что макроинструкция в шестой строке файла «example.asm», в первой строке своего определения содержит неразрешенную инструкцию.

## Синтаксис

Ориентированность на широкий спектр операционных систем (не все из которых поддерживают многобайтовые символы), ограничивает используемую в исходных текстах палитру допустимых символов до однобайтовых вариаций.

Под абстракцией «символ» воспринимается символ с привязкой к конкретному ASCII коду, а не конкретному начертанию.

В палитре символов можно выделить отдельные группы:

**Местозаполнитель между токенами:**

- ASCII код 9 (Табуляция)

- ASCII код 32 (Пробел)

**Разделитель строк:**

- ASCII код 10 (Разрыв строки)

- ASCII код 13 (Возврат каретки)

**Кавычки:**

- ASCII код 34 (Двойные кавычки «"»)

- ASCII код 39 (Одинарные кавычки «'»)

**Специальные символы:**

- ASCII код 35 (Решерка «#»)

- ASCII код 38 (Амперсанд «&»)

- ASCII код 40 (Открывающая круглая скобка «(»)

- ASCII код 41 (Закрывающая круглая скобка «)»)

- ASCII код 42 (Знак умножения «\*»)

- ASCII код 43 (Знак плюс «+»)

- ASCII код 44 (Запятая «,»)

- ASCII код 45 (Знак минус «-»)

- ASCII код 46 (Точка «.»)

- ASCII код 47 (Знак деления «/»)

- ASCII код 58 (Двоеточие «:»)

- ASCII код 59 (Точка с запятой «;»)

- ASCII код 60 (Знак меньше «<»)

- ASCII код 61 (Знак равно «=»)

- ASCII код 62 (Знак больше «>»)

- ASCII код 91 (Открывающая квадратная скобка «[»)

- ASCII код 92 (Обратный слеш «\»)

- ASCII код 93 (Закрывающая квадратная скобка «]»)

- ASCII код 96 (Апостроф «`»)

- ASCII код 123 (Открывающая фигурная скобка «{»)

- ASCII код 124 (Вертикальная черта «|»)

- ASCII код 125 (Закрывающая фигурная скобка «}»)

- ASCII код 126 (Тильда «~»)

**Запрещенный для использования в исходном тексте:**

- ASCII код 0 (Пустой символ)

**Запрещенный для использования в тексте инструкций:**

- ASCII код 26 (Подстановка «→»)

### Исходный текст

Исходный текст представляет собой последовательность строк, они собираются из строк основного файла, из строк подключаемых файлов к основному, из строк подключаемых файлов к ранее подключенным файлам и т.д.

Текст любого основного или подключаемого файла обычно представлен всем содержимым этих файлов, но он также может быть принудительно завершен первым встреченным символом с ASCII кодом 0, при этом сам символ конца текста не является частью текста, поэтому этот символ не может появиться в исходном тексте.

Каждая строка исходного текста может быть пустой или содержать текст инструкции и/или текст комментария.

### Текст комментария

Единственный предустановленный тип комментария – однострочный, каждый такой комментарий начинается с символа «;» и заканчивается разрывом строки.

### Текст инструкции

Каждая инструкция начинается с новой строки и заполняет все пространство этой строки не занятое под комментарий. Также, никакая инструкция не может закончиться символом «\» (без учета пробелов и табуляций после него), если этот символ последний в пространстве строки занятом текстом инструкции, то на его место (ставится пробел, а затем) к инструкции прикрепляется содержимое следующей строки, и так далее с неограниченным уровнем вложенности.

Над текстом каждой инструкции обязательно проводится разбор на элементы, для внутренних нужд компилятора в процессе этого разбора задействован символ с ASCII кодом 26, поэтому непосредственно использовать его в тексте инструкций нельзя.

Текст инструкции после разбора на элементы превращается в тело инструкции, которое может быть исполнено на одной из стадий компиляции.

Если к комментариям грубо причислить все, что в процессе компиляции не будет исполняться, то на уровне инструкций можно создать многострочный комментарий. Но природа такого комментария будет унаследована от инструкций, над его текстом будет произведен разбор элементов на типы, и, следовательно, будет запрещено использование внутри такого комментария символ с ASCII кодом 26.

Тело любой инструкции представляет собой последовательность элементов, каждый из которых может принадлежать одному из трех разных типов: 1 - самостоятельный символ, 2 - непрерывный символ и 3 - символ в кавычках.

Первый тип элемента инструкции – представляет собой односимвольный элемент и воспринимается в тексте инструкции как самостоятельный символ независимо от того отделен ли он от текста соседних элементов или выполнен с ними слитно. Палитра односимвольных элементов немногочисленна: «+-\*/=<>()[]{}:,|&~#`».

Второй тип – представляет собой многосимвольный элемент, последовательность символов которого в тексте инструкции начинается с символа отличного от кавычек (двойных «"» и одинарных «'»), выполнена слитно и не включает самостоятельных символов.

Третий тип – также представляет собой многосимвольный элемент, но последовательность символов которого в тексте инструкции начинается с символа кавычек (не важно двойных «"» и одинарных «'») и не заканчивается пока в пределах строки не будет встречен символ такой же кавычки, даже символ «;» будучи встреченным в пределах такой последовательности не открывает никакого комментария, а просто становится символом этой последовательности. Если после завершающей кавычки слитно стоит символ такой же кавычки, то в последовательность включается 1 символ этой кавычки, а сама последовательность продолжается до следующего такого же символа кавычек. Внешние открывающая и закрывающая кавычки не становятся частью элемента, они лишь выполняют роль индикатора для его типизации.

Примечание: какие символы не могут стать частью строки в кавычках:

- символ с ASCII кодом 0 не может появиться в исходном тексте, следовательно, не может появиться в символе в кавычках, являющимся частью исходного текста.

- символ с ASCII кодом 9 (табуляция) автоматически заменяется пробелами в IDE fasmw.exe, следовательно, не появится в символе в кавычках, но если исходный текст создан в другом редакторе, символ будет корректно обработан.

- символы разрыва строк и возврата каретки – не позволят найти в пределах строки (т.к. строка завершится этими символами) завершающего символа кавычек, что сгенерирует соответствующую ошибку.

- символ с ASCII кодом 26 не может появиться в инструкции, следовательно, не может появиться в символе в кавычках, являющимся частью инструкции.